Trabajo de Final de Grado

Grado en Ingeniería en

Tecnologías Industriales

Diseño y fabricación de un vehículo aéreo no tripulado

MEMORIA

Autores: Eduard Valentino Birau

Jorge Cantero Guerrero

Director: Juan Manuel Moreno Eguilaz

Convocatoria: Julio 2015

Escola Tècnica Superior

d'Enginyeria Industrial de Barcelona

Resumen

Este proyecto describe el proceso seguido por dos estudiantes de ingeniería para diseñar y fabricar un prototipo de un cuadricóptero *dron* de uso civil.

Se pasa por diferentes etapas y fases por las que debería pasar cualquiera para poder llevar a cabo lo anteriormente mencionado. También, en cada etapa, se intenta dejar muestras de los conocimientos adquiridos durante el Grado en Ingeniería en Tecnologías Industriales. Se aplican conocimientos de cálculo, geometría, mecánica del sólido rígido, resistencia de materiales, informática, electrónica…

Se trata de un proyecto de pensar y hacer, por lo que todo lo que se verá a continuación tiene una parte intelectual de investigación, análisis y estudio y otra parte de fabricación y llevar a cabo lo que con anterioridad se decide de manera teórica.

El proyecto empieza desde cero y acaba en el punto en que se dispone de una plataforma desde la cual se podrá desarrollar un software de vuelo robusto y dejar volar la imaginación con todas las posibilidades que quepan.

INDICE

GLOSARIO

Las referencias a ejes de coordenadas vienen claramente definidos en la Fig. 11.

q: Vector de coordenadas generalizadas.

L: Expresión de la ecuación de Lagrange.

T_{tras}: Energía cinética de traslación.

T_{rot}: Energía cinética de rotación.

U: Energía potencial.

$\mathbb{I}$: Tensor de inercia.

R_x^{θ}: Matriz de giro respecto *eje x*.

R_y^{φ}: Matriz de giro respecto *eje y*.

R_z^{ψ}: Matriz de giro respecto *eje z*.

$\mathbb{J}$: Matriz de inercia en orientación aleatoria.

θ: Ángulo de *pitch*.

φ: Ángulo de *roll*.

ψ: Ángulo de *yaw*.

τ: Vector de momentos.

M_{θ}: Momento generado en el *eje x*.

M_{φ}: Momento generado en el *eje y*.

M_{ψ}: Momento generado en el *eje z*.

T_i: Fuerza que genera el motor *i*.

K_{empuje}: Constante que relaciona la fuerza de empuje la velocidad de giro de un motor.

ω_i: Velocidad de giro de un motor.

τ_{mi}: Momento torsional de un motor al girar.

I_{zm}: Momento de inercia en el eje de giro de los motores.

τ_{dragi}: Momento de rozamiento del aire con las hélices.

INDICE DE FIGURAS

INDICE DE TABLAS

1. Introducción

El uso de vehículos aéreos no tripulados, también llamados drones, ha sufrido un gran auge durante los últimos años. Este auge se ha visto, sobretodo, en el uso de los drones para uso civil y para el entretenimiento, de manera que cada vez se han hecho más accesibles al público. Además, las tecnologías integradas han ido facilitando de tal manera el pilotaje de estos vehículos que hasta un niño es capaz de utilizar uno.

En este proyecto se ha querido recrear la dificultad de diseñar y construir un dron desde cero, con todo lo que ello supone: diseñar una estructura mecánica, trabajar con elementos de comunicación, programar, etc. De tal manera, este proyecto se trata de un proyecto de gran envergadura en el que se ha pasado por distintas etapas en las que se ha tenido que resolver los problemas que se pudiera encontrar quien desee diseñar y construir su propio dron.

Por lo que respecta a la estructura mecánica, eligiendo los materiales que mejor se adapten, cuyo mecanizado sea posible con las herramientas disponibles y con el objetivo de minimizar el peso y maximizar la resistencia. Diseñando pieza por pieza, para que todos los elementos encajen correctamente y para hacer la fabricación lo más rápida y simple posible. Y finalmente, fabricando las piezas diseñadas respetando las medidas definidas y haciendo que el proceso sea lo más simple posible para su fácil reproducción.

Por lo que respecta a la electrónica: la de potencia que incluye los actuadores, sus controladores y la batería, que deben ser compatibles entre sí y que, en conjunto, puedan permitir el vuelo del dron. La electrónica de control, que supone la elección del procesador que mejor se adapte y que pueda procesar la información requerida para un vuelo estable y de fácil pilotaje; la elección de los sensores que permitan conocer el estado del dron durante el vuelo para su posterior procesamiento; y los módulos de comunicación, que realicen un intercambio de información estable y con alcance suficiente.

Y finalmente, por lo que respecta a la programación: escribiendo un código, haciendo uso de librerías ya existentes, para el procesador, de manera que lea las entradas de los distintos módulos del dron (sistema de comunicación, sensores, etc.), los procese y los transforme en señales a enviar a los motores, para que estos funcionen de acorde a los órdenes del piloto y para que hagan el vuelo lo más estable e intuitivo posible.

Los principales motivos que han llevado a los autores de este proyecto a la elección de este tema han sido las ganas de llevar a cabo un proyecto extenso y aprender conocimientos prácticos sobre los drones y su funcionamiento. Es una oportunidad única para aplicar lo aprendido durante la carrera en el diseño y construcción de un dispositivo funcional y que en acción puede llegar a ser muy sorprendente.

Por supuesto, también se ha elegido por la diversión del proyecto, ya que aunque el diseño y la construcción requieren muchas horas de trabajo y frustraciones, el trabajo práctico de construcción y montaje y las pruebas de vuelo pueden llegar a convertirse en un auténtico y verdadero hobby.

Tal y como se expone en esta introducción de lo que supone este extenso proyecto, las horas de dedicación en el proyecto incluyen las de investigación, selección de información, diseño, fabricación y pruebas. Tantas y tan largas pueden ser estas fases que los objetivos y el alcance de este Trabajo de Final de Grado serán, como mínimo, conseguir una estructura óptima y una distribución de componentes y elementos para, en un futuro, poder dedicarse exclusivamente al diseño y desarrollo de un software de vuelo robusto. Esto es, que se pueda diseñar un programa que permita hacer volar el cuadricóptero de manera sencilla y capaz de resistir diferentes perturbaciones. Otra tema, y mucho más visual y complicado, sería poder desarrollar un software de control que permitiera al cuadricóptero volar a cierta altura, detectar obstáculos para salvarlos, poder hacer acrobacias de vuelo... lo que la imaginación dé de sí. Pero todo eso es algo pensado para realizarse en proyectos futuros.

Además, para dejar todo listo para hacer volar el *dron*, se pretende diseñar y montar una estación externa con la cual controlarlo. Para ello, se deberá elegir un hardware adecuado que satisfaga las necesidades futuras.

Sin embargo, los autores de este proyecto no se han querido cerrar puertas ni ponerse límites, por lo que durante los meses que ha durado este proyecto se han dejado la piel y han intentado avanzar todo lo que se pudiera hacia la meta final de hacer volar un cuadricóptero diseñado y fabricado por ellos mismos.

2. Antecedentes

La historia de los cuadricópteros empieza en el 1907 con los hermanos Jaques y Louis Breguet, que construyeron el "Gyroplane No1", la primera aeronave de vuelo vertical[3]. Esta estaba constituida por una estructura tubular de metal con un asiento para el piloto y una fuente de energía en la parte central. En su primer vuelo solo llegó a una altura de 0,6 metros, aunque no fue en vuelo libre, ya que cuatro hombres ayudaban, mediante cuerdas, a mantener la aeronave estable.

El segundo intento lo hizo un ingeniero francés, Etienne Oemichen, en 1924 que pilotó su cuadricóptero, el Oemichen No.2, a una distancia de 360 m con estabilidad y maniobrabilidad aceptables[4].

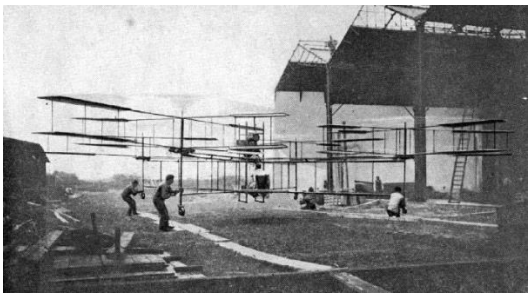

Fig. 1. Gyroplane No1. Fuente: www.famillebreguet.com.

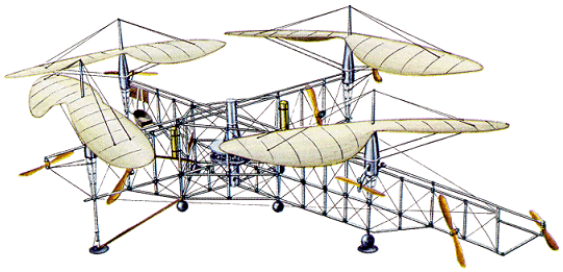

Fig. 2. Oemichen No.2 Fuente: www.aviastar.org.

Después de los primeros intentos, la idea de los cuadricópteros fue abandonada por diversos motivos:

- Al estar el motor en el centro de la estructura había que instalar correas para transmitir el par a las cuatro hélices, lo que causaba un aumento importante del peso y la realización de tareas de mantenimiento constante.
- Los cuadricópteros por su naturaleza son aeronaves inestables, por el simple hecho de que los 4 rotores son ligeramente diferentes entre ellos y para mantener un vuelo estable no es suficiente solo con hacer girar los rotores a la misma velocidad, sino que también hay que hacer ajustes para estabilizar el vuelo. Esta labor de estabilización supone una carga de trabajo demasiado grande para el piloto.

Con la aparición y desarrollo de los motores eléctricos, los microcontroladores y la tecnología MEMS (Micro Electro Mechanical Systems), el concepto de cuadricóptero ha vuelto a la luz y está teniendo un gran éxito en muchos sectores.

Actualmente, el uso de los cuadricópteros o drones es muy amplio: uno de los principales sectores que está haciendo uso de esta tecnología es el de la fotografía, ya que hay versiones de estas aeronaves muy ágiles y pequeñas, que permiten montar una cámara fotográfica y acceder a sitios de difícil acceso y tomar fotos aéreas espectaculares.

Fig. 3. Dron Draganflyer X4-P. Fuente: www.draganfly.com.

Entre otras aplicaciones se encuentran la búsqueda de personas desaparecidas, que para estas misiones son aparatos muy útiles, ya que pueden acceder a zonas montañosas o nevadas muy fácilmente y su tamaño reducido permite que cualquier estación de montaña pueda tener uno. Otra ventaja que tiene es su precio, muy reducido respeto al uso de un helicóptero de búsqueda.

Fig. 4. Fotografía de dos personas perdidas en la montaña tomada por un dron. Fuente: www.iuavs.com.

Una posible situación de búsqueda podría ser la representada en la Fig. 4, en la cual el mismo software del *dron* reconoce la presencia de las víctimas y puede informar de la posición de estas, mediante posicionamiento por GPS, al centro de control para enviar equipos de rescate.

Otra aplicación parecida es la de prevención y control de incendios, ya que en épocas y zonas de alto riesgo se puede establecer supervisión constante para poder así reaccionar lo más rápido posible y evitar que personas resulten heridas y también que se pierdan hectáreas de bosque.

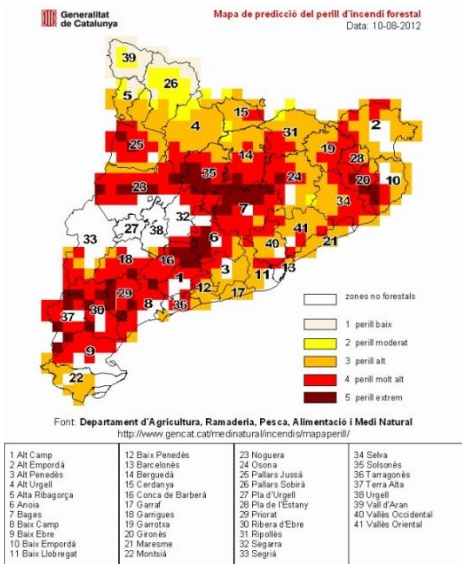

Fig. 5. Mapa predictivo del peligro de incendios forestales. Fuente: sección de noticias de la página web de TV3.

Un ejemplo de esta aplicación está en el *dron* español anti-incendios FT-Altea [5], diseñado por la empresa Flightech Systems. Este dron, de 80 kilos de peso y seis metros de envergadura, permite realizar operaciones de vigilancia mediante varias cámaras con las que está dotado, entre las cuales se encuentra una cámara térmica que permite identificar los focos iniciales de los incendios. También la labor del *dron* será de enviar información sobre el estado del viento y otras condiciones climatológicas para ayudar a los bomberos en la contención de los incendios.

Fig. 6. Un avión no tripulado del modelo FT-Altea diseñado para la vigilancia de incendios. Empresa fabricante *Flightech Systems*. Fuente: www.elmundo.es.

Otros campos de aplicación de *drones* podrían llegar a ser el sector de la agricultura, para controlar y monitorizar el estado de cultivos, la geología para la realización de mapas y la exploración de cuevas y precipicios, la monitorización del tráfico de coches y muchos otros más (Amazon tiene previsto utilizar drones para el reparto de los bienes vendidos a través de internet).

El vehículo descrito en este proyecto pertenece al campo de los drones para uso de entretenimiento, ya que para los autores es la manera más simple de iniciarse en el mundo de los *drones* y en el cual se tienen más probabilidades de obtener buenos resultados.

Actualmente el uso de los *drones* como hobby se está extendiendo y con ello también las empresas que fabrican este tipo de aparatos. El rango de precios va desde menos de 100€[6], para cuadricópteros pequeños y fáciles de maniobrar que sirven para la iniciación y para aprender su funcionamiento, hasta varios miles euros[7], para aquellos que están provistos de cámaras para realizar fotos y videos, implementaciones muy interesantes en el software de gobierno y muchos otros complementos. A continuación se nombran algunos modelos presentes en el mercado y sus características:

- Parrot AR Drone 2.0

  El AR Drone es el primer dron controlado mediante conexión WiFi. Es un cuadricóptero de gama media que permite realizar vuelos de hasta 12 minutos. Tiene un peso de 420 gramos y está provisto de sensores de movimiento, como por ejemplo un acelerómetro, un giroscopio y un altímetro para mantener un vuelo estable. La aplicación que viene con el aparato permite controlar el dron desde un Smartphone y mediante la cámara que lleva incorporada se pueden ver en directo las imágenes registradas. Tiene un precio aproximado de 300€.

Fig. 7. Parrot AR Drone 2.0. Fuente: www.parrot.com

- DJI Phantom 2

El DJI Phantom 2 es un dron especialmente diseñado para sacar fotos aéreas de alta calidad. Tiene un peso de 1Kg y tiene una autonomía de hasta 25 minutos. El control se realiza mediante un mando de radiocontrol, pero se puede utilizar un Smartphone para ver las imágenes de las cámaras. El software está provisto de varias modalidades de vuelo, entre ellas una opción interesante es la de "Return Home", la cual permite que el cuadricóptero, si la distancia entre este y el piloto ha excedido el alcance del mando radiocontrol, vuelva al punto donde ha despegado y aterrice de manera automática. Su precio oscila en torno a los 1000€.

Fig. 8. DJI Phantom 2. Fuente: www.dji.com.

- Mikrokopter L4-ME

Fig. 9. Mikrokopter L4-ME. Fuente: www.mikrokopter.de.

A diferencia de los modelos presentados en los apartados anteriores, este cuadricóptero se vende en forma de kit de montaje, lo que significa que el usuario necesitará tener un mínimo de habilidades técnicas para poder realizar el ensamblaje de las piezas. Pasar por el proceso de construir el dron es muy útil, ya que confiere conocimientos sobre el funcionamiento de este, lo que permite hacer futuras mejoras sin necesidad de volver a comprar todos los componentes.

En el kit se incluye la estructura, los motores y la placa controladora, provista de un software abierto de control, todo por un precio de 600€.

Al ser el software abierto, este recibe continuamente actualizaciones y mejoras sin coste alguno para el usuario.

- Drone Devo QR X350 Standard

El Drone Devo QR X350 Standard es un modelo fácilmente transportable, de manera que se puede desmontar y montar en poco tiempo. Tiene, además, una plataforma para incorporar cámaras compactas y está especialmente diseñado para la cámara GoPro Hero 3. Consta de un sistema de comunicación por radiofrecuencia y está equipado con sensor GPS, lo que permite la función "Return Home". Tiene una autonomía de entre 10 y 15 minutos en función, claro está, de la cámara que se incorpore. Se vende por 440€.

Fig. 10. Drone Devo QR X350 Standard. Fuente: www.walkera.com.

A continuación se presenta una tabla resumen con las características de los *drones* que actualmente se comercializan para aficionados. Dicha tabla ha sido una guía orientativa para la elección posterior de los diferentes componentes del *dron* propuesto, cuyo diseño y fabricación es objeto este proyecto.

	Parrot AR DRONE 2.0	DJI Phantom 2	Mikrokopter L4-ME	Devo X350
Dimensiones (cm)	51,5x51,5	35x35	28,3x28,3	28,9x28,9
Peso (g)	420	1000	~800	1000
Batería	3S	3S	4S	3S
Carga (mAh)	1000	5200	2200	5200
Tensión (V)	11,1	11,1	14,7	11,1
Capacidad descarga	10C	-	20C	-
Potencia motor (W)	15	140	110	-
Kv	1700 (inrunner)	920	760	-
Imax	24A	17A	10A	15A
Hélices (in)	especiales parrot	10x4.5	10x4.5	-
flight time (min)	12	20	22	25
Comunicaciones	WiFi (50m-120m)	RF	RF	RF
Acelerómetro	Sí	Sí	Sí	Sí
Giroscopio	Sí	Sí	Sí	Sí
Altímetro	Sí	Sí	Sí	Sí
GPS	NO	Sí	Posible	Sí
Accesorio	CÁMARA	CÁMARA	Posible cámara	Posible cámara
Precio (€)	300	549 – 1199	586	300 - 1016

Tabla 1. Resumen del análisis de diferentes *drones* existentes en el mercado (Fuente: propia)

3. Dinámica de vuelo

La teoría que explica cómo vuela un *dron* o cuadricóptero es sencilla de explicar y entender: del cuadricóptero, lo único que se puede controlar y que actúa con el medio son los motores con sus respectivas hélices. Estos motores están fijos a la estructura del *dron* (existen modelos en los que los motores son móviles respecto a su carcasa), de manera que la única manera de actuar con el medio es con la velocidad de giro de los motores y el sentido de rotación de los mismos.

Antes de nada, hay que definir qué variables son las que describen el vuelo del *dron*. En este proyecto las coordenadas absolutas de altitud y posicionamiento en el espacio no se han tenido en cuenta porque no se han considerado importantes en el desarrollo principal del software de vuelo.

Por lo tanto, las variables que definen el estado de vuelo del *dron* serán los principales ángulos de Euler: cabeceo, alabeo y guiñada; pitch, roll y yaw en inglés (a partir de ahora se denominarán en inglés por su uso más extendido).

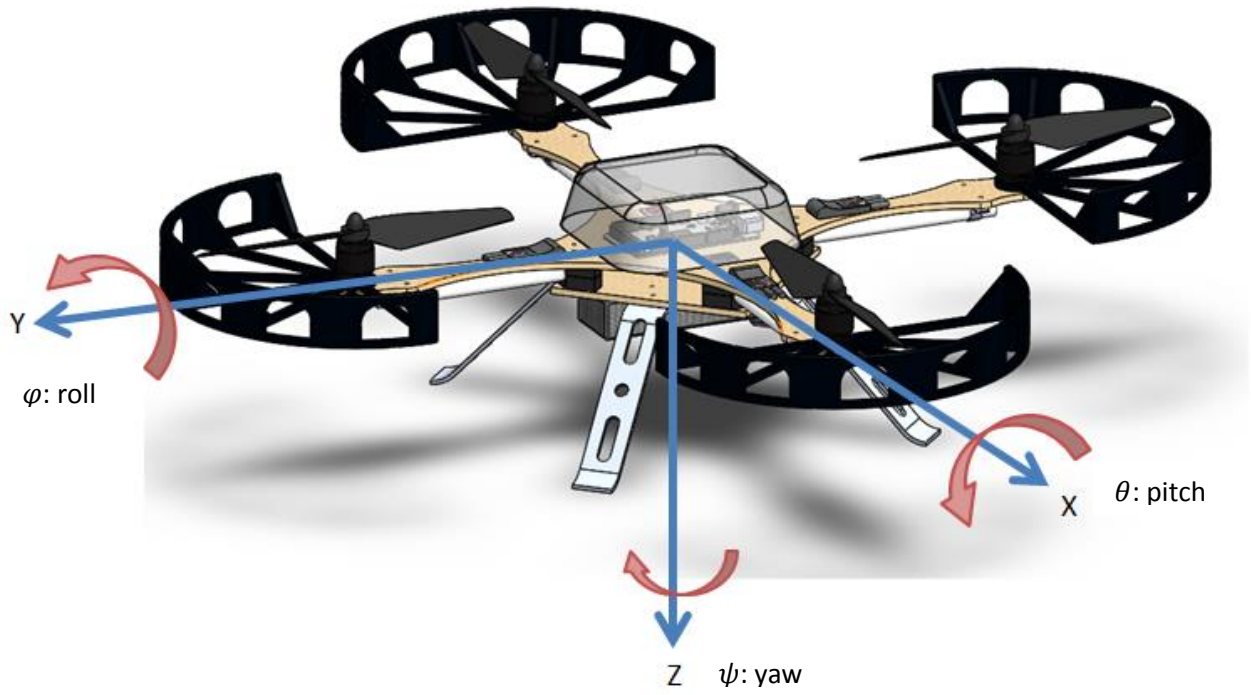

Fig. 11. Ejes de coordenadas del cuadricóptero (Fuente: propia)

Una vez se han identificado los ejes de coordenadas y las coordenadas con las cuales se define el movimiento del *dron,* se procede a su estudio dinámico y estático.

3.1. Definición

En este proyecto se ha numerado cada motor para su fácil identificación: el motor número uno es el que corresponde al eje positivo *y*; el dos, al eje negativo *x*; el tres, al eje negativo *y*; finalmente, y de manera ya obvia, el motor número cuatro corresponde al eje positivo x.

Así, se pueden explicar los diferentes estados del *dron,* definiendo la consecución de tres movimientos:

3.1.1. Movimiento de pitch

Para conseguir una variación positiva del ángulo de pitch se le administra más potencia al motor número tres y menos al motor número uno. Es decir, se aumenta las revoluciones del motor tres y se disminuyen, en la misma medida, las del motor uno. De esta manera se mantiene la potencia del conjunto y se consigue un momento en el eje *x* que hace girar el *dron* en sentido positivo, tal como se ha definido. Además, se consigue que el cuadricóptero avance en el *eje y positivo* a causa de la descomposición de las componentes de las fuerzas del plano. Para conseguir una variación negativa y un retroceso se aplica el diferencial de potencia en sentido contrario.

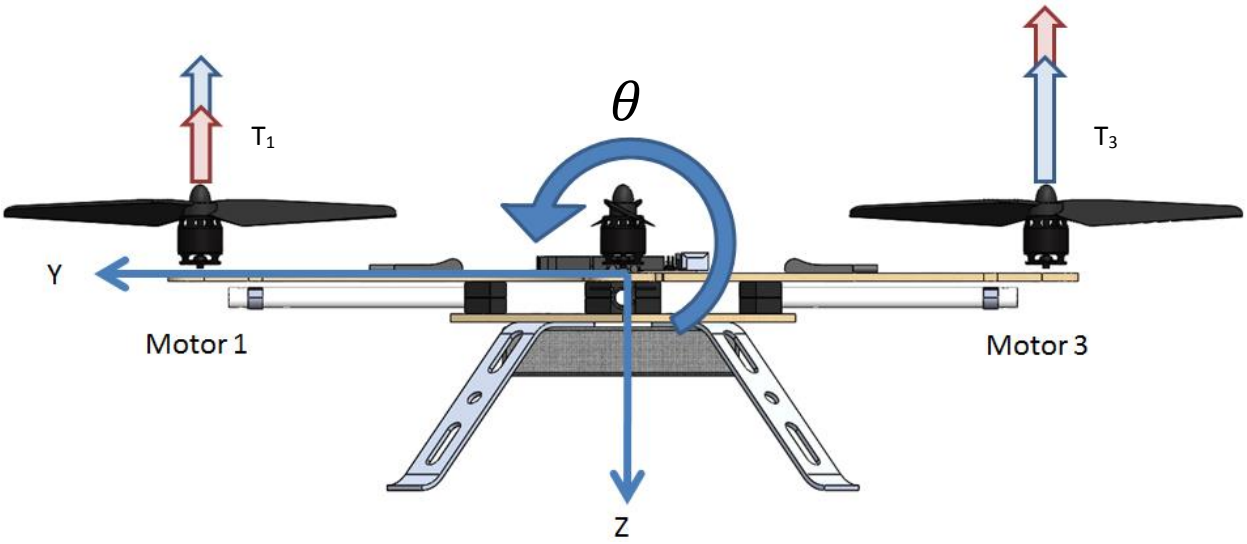

Fig. 12. Rotación positiva del *eje X*, ángulo de pitch (Fuente: propia)

Tal como se puede observar en la Fig. 12, se representa en una primera instancia las fuerzas de los motores en color azul y después, en rojo, las fuerzas con el respectivo diferencial. Tras el análisis dinámico de este estado, se pasa a la siguiente posición, la cual induce al avance.

Según el Teorema de la Cantidad del Movimiento (TQM a partir de ahora) se tiene que, en el eje *z*, las fuerzas han de compensar el peso, para que el dron se mantenga estable y a una cierta altura. En el eje y, en este estado, no existen componentes de las fuerzas de los motores.

Ecuación de equilibrio eje *z*:

$$\sum_{i=1}^{4} T_i = mg \qquad \text{(Ec. 1)}$$

Esto no introduce ninguna variación en el estado del *dron* si el diferencial que se añade en el motor 3 es el mismo que se extrae en el motor 1. En cambio, según el Teorema del Momento Cinético aplicado al centro de masas se tiene que (como aproximación, se considera que el centro de masas se encuentra en la intersección de los tres ejes anteriormente definidos):

$$\dot{\overline{GK}} = \sum \overline{GP} \wedge \overline{F_{ext}}(P) \qquad \text{(Ec. 2)}$$

$$\mathbb{I}_G\dot{\Omega} = \begin{bmatrix}0\\d\\0\end{bmatrix} \wedge \begin{bmatrix}0\\0\\-T_1\end{bmatrix} + \begin{bmatrix}-d\\0\\0\end{bmatrix} \wedge \begin{bmatrix}0\\0\\-T_2\end{bmatrix} + \begin{bmatrix}0\\-d\\0\end{bmatrix} \wedge \begin{bmatrix}0\\0\\-T_3\end{bmatrix} + \begin{bmatrix}d\\0\\0\end{bmatrix} \wedge \begin{bmatrix}0\\0\\-T_4\end{bmatrix} \quad \text{(Ec. 3)}$$

$$\begin{bmatrix}I_{xx} & 0 & 0\\0 & I_{yy} & 0\\0 & 00 & I_{zz}\end{bmatrix}\begin{bmatrix}\ddot{\theta}\\\ddot{\varphi}\\\ddot{\psi}\end{bmatrix} = \begin{bmatrix}-dT_1\\0\\0\end{bmatrix} + \begin{bmatrix}0\\-dT_2\\0\end{bmatrix} + \begin{bmatrix}dT_3\\0\\0\end{bmatrix} + \begin{bmatrix}0\\dT_4\\0\end{bmatrix} = \begin{bmatrix}d(T_3 - T_1)\\d(T_4 - T_2)\\0\end{bmatrix} \quad \text{(Ec. 4)}$$

Tal como se puede observar en la ecuación (4) los momentos en el eje *y* desaparecen al ser T_4 igual que T_2. Por otra parte, se crea un diferencial de momentos en el eje *x*, el cual provoca la aparición de una aceleración angular en ese eje, dando pie al movimiento anteriormente explicado. La expresión de la aceleración angular que aparece es la siguiente.

$$\ddot{\theta} = \frac{d(T_3 - T_1)}{I_{xx}} \quad \text{(Ec. 5)}$$

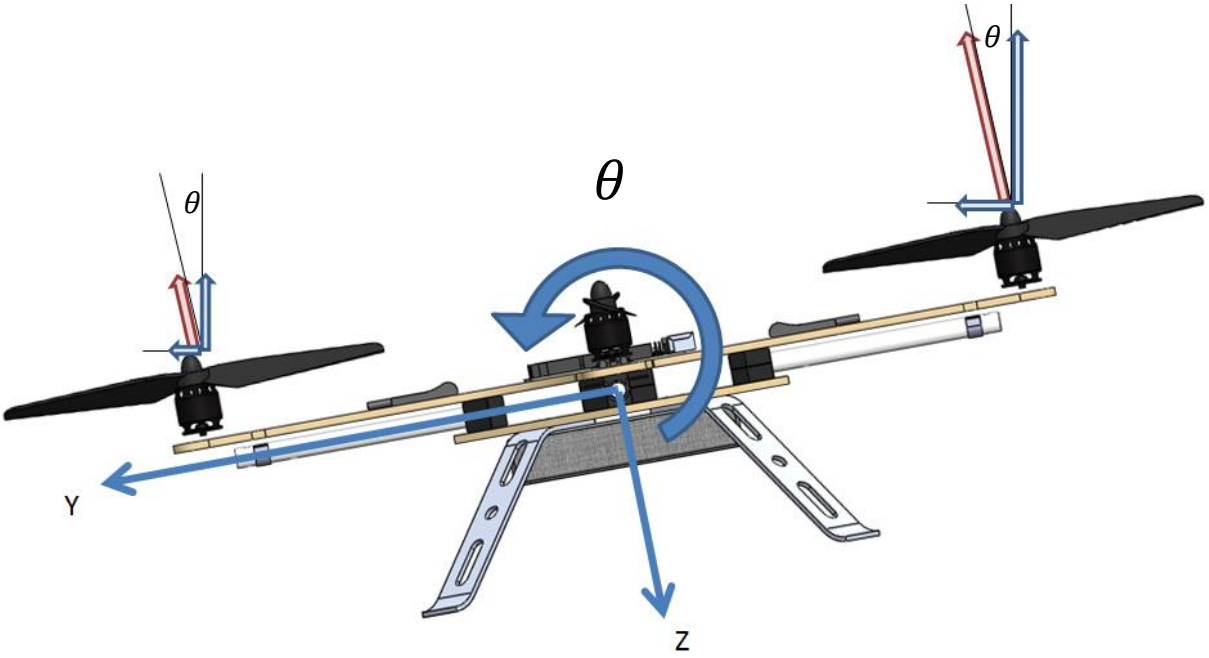

Fig. 13. Avance inducido por el ángulo de pitch (Fuente: propia)

3.1.2. Movimiento de roll

Para conseguir una variación positiva del ángulo de roll se le administra más potencia al motor número cuatro y menos al motor número dos. El funcionamiento es prácticamente idéntico al movimiento de pitch. En la Fig. 14 se demuestran las similitudes.

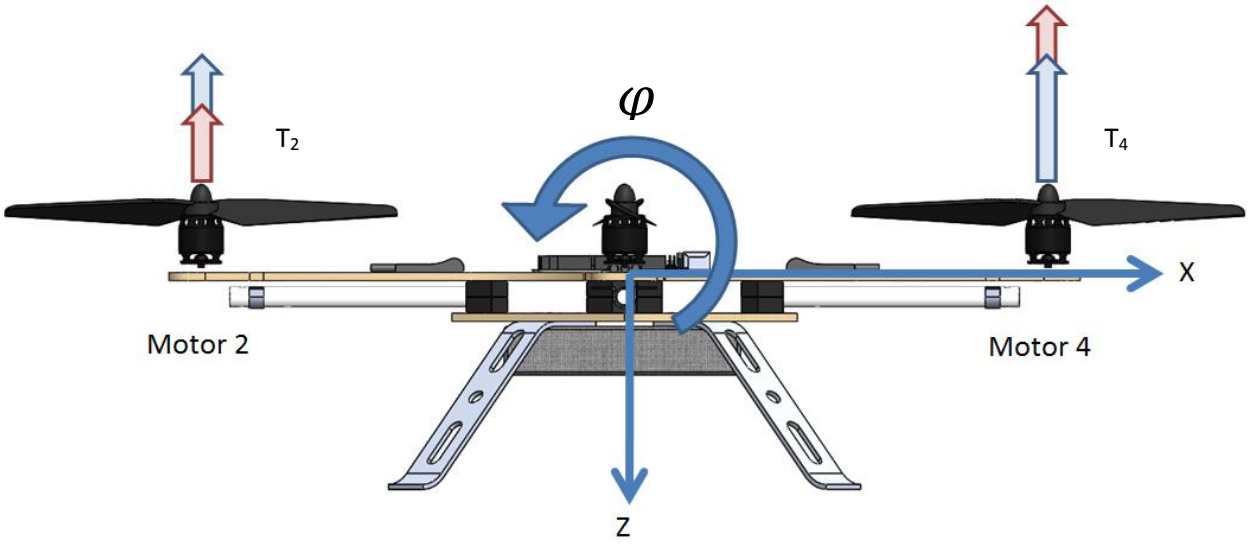

Fig. 14. Rotación positiva del *eje Y*, ángulo de roll (Fuente: propia)

En este caso también se cumple la ecuación 1 de equilibrio del TQM y, al aplicar el TMC, se llega a la ecuación número 4. Esta vez, las fuerzas que se anulan para crear momento son T_1 y T_3, mientras que al aparecer una diferencia entre T_2 y T_4 se genera un momento en el eje *y* positivo cuya aceleración angular queda definida por la ecuación 6:

$$\ddot{\varphi} = \frac{d(T_4 - T_2)}{I_{yy}} \qquad \text{(Ec. 6)}$$

De igual manera, aparece una componente horizontal que genera el movimiento de traslación en el sentido en el que el *dron* se ha inclinado.

3.1.3. Movimiento de yaw

El movimiento de yaw es distinto al de pitch y roll, en el sentido que no genera una traslación. Se basa en la variación de giro de los motores (como todos los movimientos) pero esta vez se actúa sobre los cuatro motores en lugar de actuar en dos de ellos.

Primero se define un sentido de giro a cada motor. En este proyecto los motores uno y tres giran en sentido horario y los motores dos y cuatro en sentido anti horario. Esto es así porque los momentos rotacionales que generan los motores se han de anular cuando giran a la misma velocidad. Por ejemplo, en caso de que los cuatro motores giraran en sentido horario nos encontraríamos que el cuadricóptero giraría en sentido anti horario en el eje *z*.

Tal como se ha definido el sentido de rotación de los motores, para obtener una cambio de orientación positivo en el eje *z* los motores dos y cuatro han de girar más rápido que los motores uno y tres. Tal movimiento corresponde a la Fig. 16, dónde ψ es positivo y, visto desde arriba, en sentido horario.

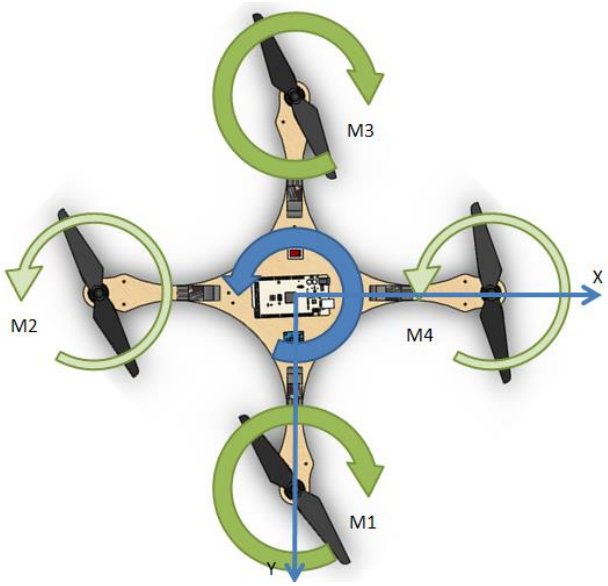

Fig. 15. Rotación negativa del *eje Z*, ángulo de yaw. (Fuente: propia)

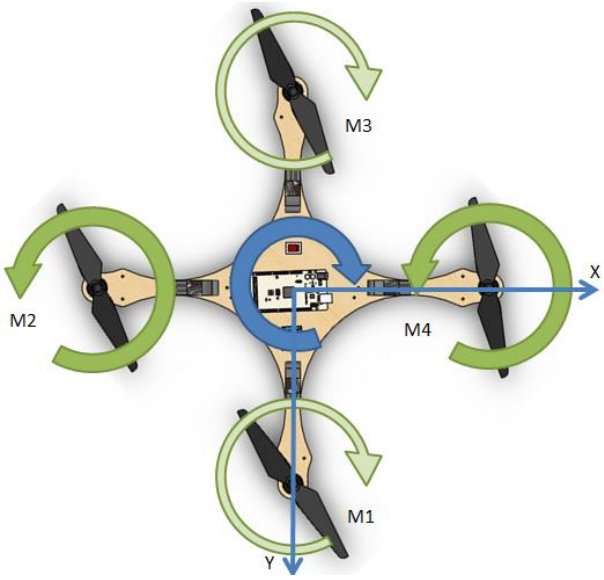

Fig. 16. Rotación positiva del *eje Z*, ángulo de yaw. (Fuente: propia)

El momento que induce este movimiento se puede expresar según la siguiente ecuación:

$$M_\psi = \sum_{i=1}^{4} \tau_{mi} = \sum_{i=1}^{4} (I_{zm} \cdot \dot{\omega}_i - \tau_{dragi}) \quad \text{(Ec. 7)}$$

El momento de yaw se produce, tal como se ha comentado antes, al sumar los cuatro momentos rotacionales de los cuatro motores. Al girar dos en un sentido y los otros dos en otro sentido, en régimen estacionario se equilibran y mantienen el dron estable en cuanto al movimiento de yaw.

En cambio, al aumentar la velocidad de giro de los motores pares y disminuir de la misma manera la de los motores impares, se crea dicho momento en el mismo sentido que los motores cuya velocidad de giro se disminuye. Esto sucede con tal de conservar la cantidad de movimiento. En este caso, el *dron* se orientaría en sentido anti horario si se considera el eje *z* positivo cuando apunta hacia el suelo u horario si se observan las imágenes.

Si se deseara un sentido de giro contrario, se debería disminuir la velocidad angular de los motores pares y aumentar la de los impares, para que el cuadricóptero girara en el sentido de los motores pares.

3.2. Modelo dinámico del *dron*

Una vez se ha descrito la dinámica de vuelo del cuadricóptero, surge la necesidad de expresar matemáticamente dicha dinámica. El objetivo de este apartado del proyecto es el conseguir un modelo que se ajuste lo máximo posible a la realidad para, posteriormente, poder realizar simulaciones y diseñar el controlador para conseguir hacer volar el *dron*. El análisis se ha realizado a partir de investigaciones y deducciones hechas por otras personas [1][2] y se ha adaptado, paso a paso, a lo que se necesitaba en este proyecto.

Para encontrar las ecuaciones de movimiento del cuadricóptero se procede por el método de Euler-Lagrange. Primero se definen las coordenadas generalizadas, las cuales nos describirán el movimiento

del *dron*. Estas coordenadas ya se han definido anteriormente y son las que corresponden a los ángulos de pitch, roll y yaw.

$$q = (\theta, \varphi, \psi) \quad \text{(Ec. 8)}$$

Para obtener las ecuaciones del movimiento se deriva la ecuación de Euler-Lagrange que une los diferentes tipos de energía que puede tener el *dron* (o cualquier sólido rígido en movimiento): energía de traslación, de rotación y/o potencial. Las energías de traslación y potencial se han dejado fuera del alcance del estudio dinámico, ya que las coordenadas que las definen no forman parte del estudio dinámico. De esta manera, la ecuación 9 queda simplificada y lleva a la ecuación 10.

$$L(q, \dot{q}) = T_{tras} + T_{rot} + U \sim T_{rot} = \frac{1}{2}\dot{q^t}\mathbb{I}\dot{q} \quad \text{(Ec. 9)}$$

Siendo $\mathbb{J}$ la matriz de inercia para cualquier orientación de los ángulos de Euler, la ecuación de Lagrange queda:

$$L(q, \dot{q}) = \frac{1}{2}\dot{q^t}\mathbb{J}\dot{q} \quad \text{(Ec. 10)}$$

Se procede a operar la ecuación de Euler-Lagrange para encontrar la expresión del movimiento. El cálculo y las operaciones sobre la ecuación 10 será igual a los momentos que aparecen en el *dron*. Las fuerzas no aparecen, ya que no se introducen con las coordenadas con las que se trabaja en este proyecto. También se han despreciado las componentes de Coriollis, ya que se modela para pequeña señal, es decir, para variaciones pequeñas de los ángulos. Se ha modelado de dicha forma porque, previendo la dificultad de controlar el vuelo, se ha preferido trabajar entorno la posición estable de coordenadas igualadas a cero. Como resultado se ha obtenido la ecuación 12, la cual proporciona toda la información de la dinámica del cuadricóptero.

$$\frac{d}{dt}\left[\frac{\partial L_{rot}}{\partial \dot{q}}\right] + \frac{\partial L_{rot}}{\partial q} = \tau \quad \text{(Ec. 11)}$$

$$\mathbb{J} \cdot \ddot{q} = \tau \quad \text{(Ec. 12)}$$

A continuación se expresarán los factores de la ecuación anterior de manera simplificada y deducida. Para conocer cómo se ha deducido su expresión y desarrollo consulte el primer capítulo del anexo.

$$\mathbb{J} = \begin{bmatrix} I_{xx} & (I_{yy} - I_{xx})\psi & (I_{xx} - I_{zz})\varphi \\ (I_{yy} - I_{xx})\psi & I_{yy} & (I_{zz} - I_{yy})\theta \\ (I_{xx} - I_{zz})\varphi & (I_{zz} - I_{yy})\theta & I_{zz} \end{bmatrix} \quad \text{(Ec. 13)}$$

$$\ddot{q} = \begin{bmatrix} \ddot{\theta} \\ \ddot{\varphi} \\ \ddot{\psi} \end{bmatrix} \quad \text{(Ec. 14)}$$

$$\tau = \begin{bmatrix} M_\theta \\ M_\varphi \\ M_\psi \end{bmatrix} = \begin{bmatrix} (T_3 - T_1) \cdot d \\ (T_4 - T_2) \cdot d \\ -\frac{K_{drag}}{K_{empuje}} \cdot \sum_{i=1}^{4} T_i \end{bmatrix} \quad \text{(Ec. 15)}$$

Este modelo ha sido introducido en el programa de simulación de Matlab/Simulink para su verificación y para conseguir encontrar los valores de los parámetros del controlador que permita ejercer de manera estable un vuelo con el cuadricóptero.

3.2.1. Implementación en Simulink

Como ya se ha comentado anteriormente, el modelo dinámico del dron se introduce en el programa de simulación Simulink del paquete Matlab. Trabaja en un entorno gráfico, con diagramas de bloques para la simulación y el diseño basado en modelos. Ofrece un editor gráfico, bibliotecas de bloques personalizables y "solvers" para modelar y simular sistemas dinámicos. Se integra con Matlab, lo que permite incorporar algoritmos en los modelos y exportar los resultados de la simulación a Matlab para llevar a cabo análisis posteriores. Con esta breve descripción se explica a groso modo los diferentes pasos a realizar para poder simular el comportamiento de un sistema.

Lo primero que se debe hacer para simular el comportamiento de un sistema es introducir en la plataforma Simulink el diagrama de bloques que lo describe. Normalmente, al realizar el análisis de variables de un sistema sencillo, como puede ser la posición vertical de un péndulo, con una sola ecuación queda definido el movimiento. Lo que resulta es una ecuación con la variable de estudio expresada en su forma temporal y su segunda derivada, la cual, una vez realizada la transformada de Laplace y trabajando en el dominio *s*, nos permite trabajar con una variable de entrada y otra de salida. Sin embargo, en este caso se tiene que las variables están conectadas entre sí y dependen unas de las otras. El diagrama de bloques pensado para describir la dinámica del *dron*, teniendo en cuenta que no se puede realizar un estudio por separado de cada variable, se puede ver en la Fig. 17.

Las variables de entrada son los tres ángulos que definen el estado del cuadricóptero: pitch, roll y yaw. Se han nombrado con el nombre griego correspondiente a la letra que los describe en el modelo dinámico: theta, phi y psi, respectivamente.

A continuación pasan por un comparador. Este comparador mide la diferencia que hay entre la consigna que se desea con el valor actual de la variable. En este caso compara la orden de accionamiento (comandada con un controlador a distancia) con la posición actual, medida con un sensor. El resultado que sale del comparador es un error, medido en grados, de la orientación del cuadricóptero.

Este error se ha querido hacer pasar por un controlador, en un inicio PID, para poder llevar las variables al valor deseado. Como hay tres variables error, hay tres controladores PID, cada uno correspondiendo a cada variable. Sin embargo, la salida de cada PID actúa de manera independiente sobre el motor que afecta a la variable de manera directa. Por ejemplo, se puede ver como el PID del pitch, llamado en el diagrama como PID theta, tiene una salida que suma en el motor 3 y resta en el motor 1. Esto sucede para conseguir llevar el *dron* a los valores deseados de la variable. Si el error es positivo, significa que la consigna es mayor que el valor actual en el que se encuentra. Para llegar a dicho valor, se debe suministrar más potencia en el motor 3 y menos en el motor 1, tal como se ha visto durante la descripción de la dinámica del dron.

El siguiente paso es pasar por un modelo aproximado de los motores. Las unidades en la salida de los controladores son PWM, correspondientes a la entrada de los motores. La salida de estos son fuerzas de empuje en Newtons, fuerzas que definen el estado del cuadricóptero.

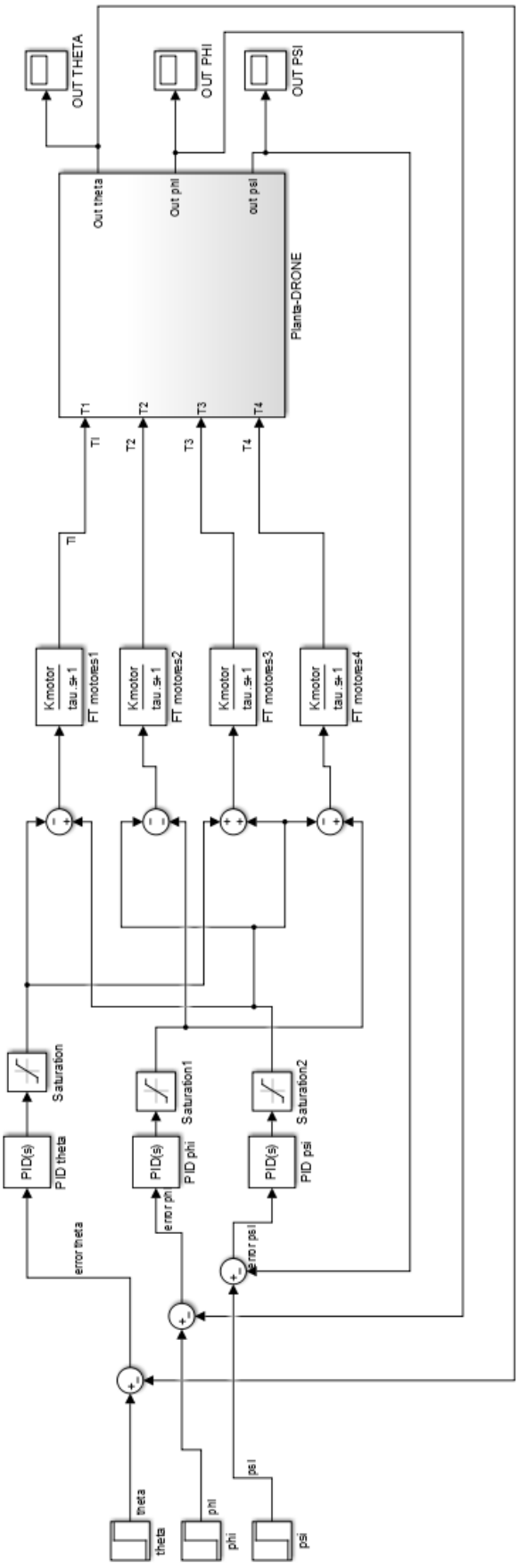

Fig. 17. Diagrama de bloques del sistema cuadricóptero (Fuente: propia)

Se ha elegido un sistema de primer orden por su sencillez y porque suelen ser los más comunes a la hora de definir el funcionamiento de motores. Lo que hay que saber de este modelo son los siguientes dos factores: la ganancia y la constante de tiempo. Tal como se ha visto en otros modelos dinámicos de *drones*, se supone una constante de tiempo de 0,1 segundos. Para conocer la ganancia del motor se realizaron unas pruebas y ensayos en las que se suministraban señales PWM a los motores y se medía la fuerza de empuje que realizaban. Los datos, gráficas y resultados se encuentran en el capítulo 5, dónde se explican las características de los motores. Al final, el valor obtenido de la ganancia de los motores es de 0,0047 *N/pwm*.

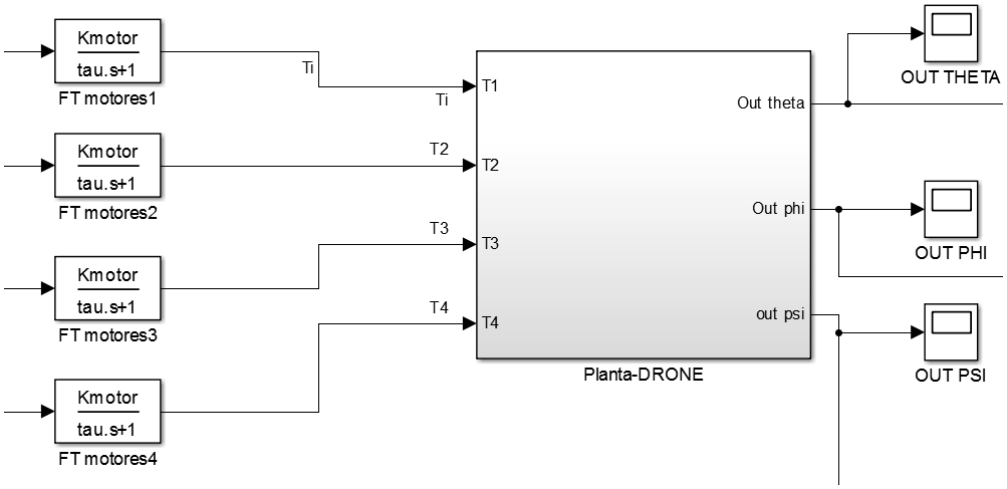

Fig. 18. "Caja negra" del modelo (Fuente: propia)

En este momento se tienen las fuerzas de los 4 motores y son las que, como ya se ha visto, definen el estado del cuadricóptero. Existe entonces una especie de "caja negra" en la cual entran 4 variables, en este caso las fuerzas de los 4 motores, y salen 3 magnitudes, los 3 ángulos que definen la orientación y, por lo tanto, lo que hace el *dron*. Esta "caja negra" no es más que el modelo dinámico que con anterioridad se ha desarrollado. También se ha implementado como un diagrama de bloques, un poco más complejo, pero al fin y al cabo las matemáticas no son más que la sucesión de operaciones entre valores y variables. Se puede ver el modelo dinámico en la Fig. 21.

Como se puede observar, se trata de la implementación de la ecuación 15. En dicha ecuación entran en juego diferentes parámetros, introducidos como constantes o ganancias. Estos parámetros son los momentos de inercia, constantes dinámicas y distancias. En la siguiente tabla quedan recogidos sus valores. En el capítulo 5 se puede encontrar los ensayos y pruebas para encontrar los valores de la constante torsional, constante de empuje y ganancia del motor. El valor de la constante de tiempo se ha tomado un valor estándar de 0,1 segundo ya que se precisa de instrumentación compleja de instalar en este tipo de motor para determinarla. Los valores inerciales son los obtenidos mediante la operación de "*Propiedades físicas*" del programa de diseño gráfico SolidWorks.

Como se puede observar en el diagrama de bloques de la Fig. 21, primero se procede a calcular los momentos que inducen cada ángulo. Este primer cálculo se implementa de manera sencilla, ya que sólo se ha de multiplicar por la distancia al centro de gravedad y por un factor entre las constantes de empuje y de torsión.

PARÁMETRO	VALOR	UNIDADES
I_{XX}; momento de inercia	0,00372532384	$\left[{Kg}/{m^2}\right]$
I_{YY}; momento de inercia	0,00434160699	$\left[{Kg}/{m^2}\right]$
I_{ZZ}; momento de inercia	0,00642186493	$\left[{Kg}/{m^2}\right]$
d; distancia motores-cdg	0,26	$[m]$
K_{drag}; constante torsional	0,0000003	$\left[{N\,m}/{\left({rad}/{s}\right)^2}\right]$
K_{empuje}; constante de empuje	0,000009	$\left[{N}/{\left({rad}/{s}\right)^2}\right]$
tau; constante de tiempo	0,1	$[s]$
K_{momtor}; ganancia motores	0,0047	$\left[{N}/{pwm}\right]$

Tabla 2. Parámetros del modelo (Fuente: propia)

Una vez se tienen los momentos, aparecen factores más complejos (a excepción de las inversas de las inercias). Estos factores salen de otra "caja negra", que no es más que una función de Matlab, un M-file, en el cual se indican las variables de entrada y las variables de salida. Las variables de entrada, o factores a operar, son las inercias, mientras que las salidas son las componentes de la inversa de la matriz $\mathbb{J}$. Tal y como se ha explicado anteriormente, las variables de control correspondientes a la orientación del *dron* están relacionadas entre sí. Para poder implementar este hecho, también han sido introducidas como entradas en el bloque correspondiente a la función de Matlab. Lo que hay escrito dentro del bloque es lo siguiente:

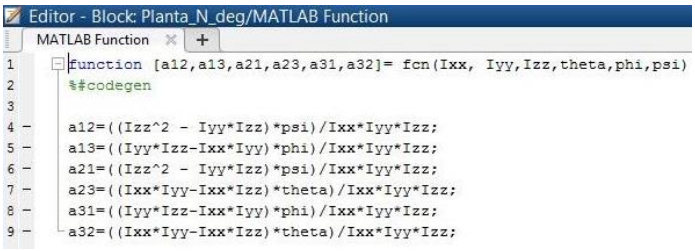

Fig. 19. Funciones incluidas en el bloque MATLAB function (Fuente: propia)

Se trata de la siguiente manera. Cómo se llega a la inversa de la matriz $\mathbb{J}$ se explica en el capítulo 1 del anexo.

$$\mathbb{J}^{-1} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad \text{(Ec. 16)}$$

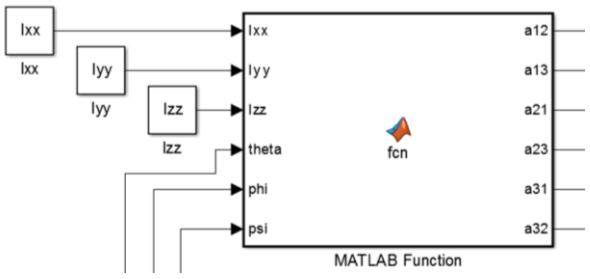

Fig. 20. Cuadro de la función de Matlab (Fuente: propia)

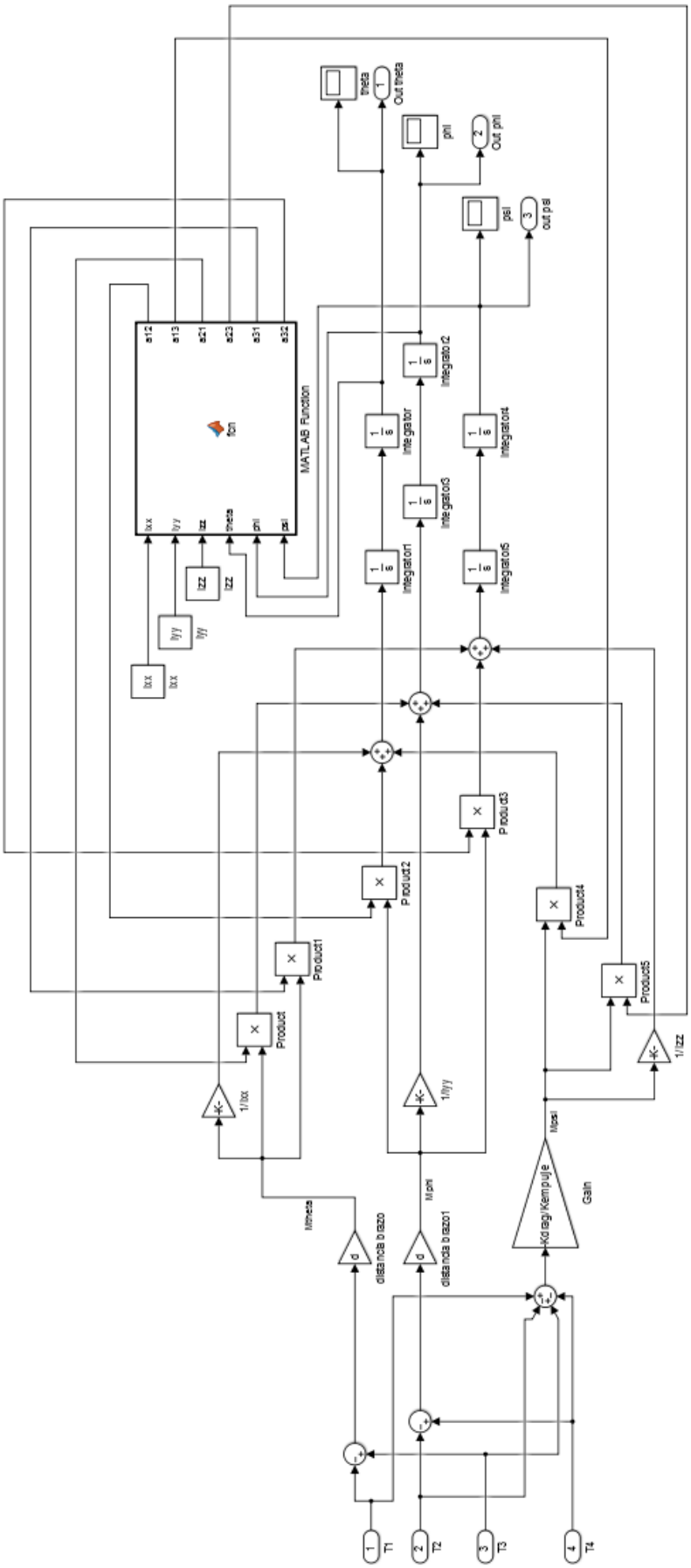

Fig. 21. Diagrama de bloques para implementar la Ec.15 (Fuente: propia)

3.2.2. Simulaciones con Simulink

Una vez se ha implementado el modelo dinámico en el programa de simulación se procede a la realización de simulaciones, para primero comprobar que lo deducido es correcto y para después poder encontrar de manera numérica los parámetros de control deseados.

Primero se realizaron simulaciones correspondientes a la planta del dron en lazo abierto, sin tener en cuenta ni la realimentación ni los sensores. Las variables de entrada son las fuerzas que ejercen cada motor y las variables de salida los ángulos que determinan la orientación.

Pulso en un motor

Para simular la variación en un motor de la fuerza que genera, se dispusieron 4 fuerzas equivalentes a 3,68*N*, lo que correspondería a la suma total de 14,72*N*, el peso total del cuadricóptero en equilibrio. A continuación se añade un pulso pequeño sumado a dicho valor en el motor 1. Los valores característicos del pulso se pueden observar en la Fig. 22 y su amplitud corresponde a un incremento de 20 milisegundos el ancho de pulso que reciben los motores en ese régimen de vuelo.

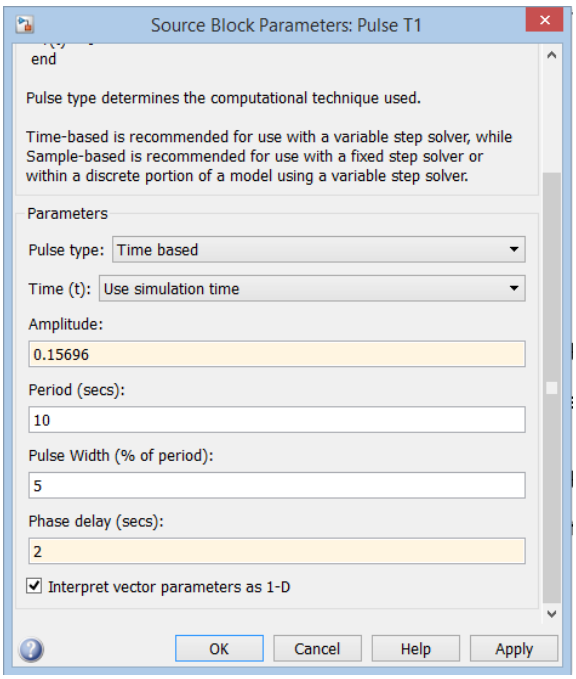

Fig. 22. Características del pulso (Fuente: propia)

El resultado esperado es que, al modificar el estado de equilibrio en el que se encuentra el *dron*, se inicie un movimiento en el ángulo de pitch. Este ángulo deberá ser negativo, y al corresponder a la doble integral de la aceleración, crecerá como lo hace un polinomio de segundo grado. No se detiene dicho movimiento porque el nuevo estado de equilibrio es un estado de giro continuo.

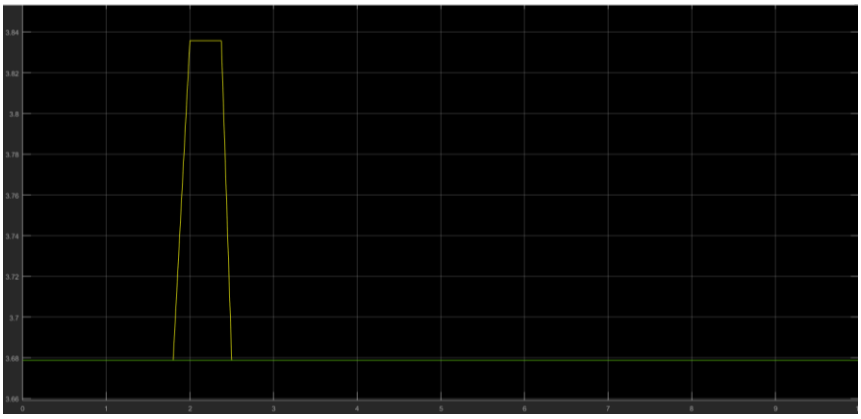

Fig. 23. Pulso positivo en el motor 1 (Fuente: propia)

En la Fig. 24, imagen correspondiente al ángulo theta, es decir, el ángulo de pitch, se puede observar en color amarillo la aceleración, en color morado la velocidad y en color turquesa la posición. Las unidades de la izquierda corresponden a radianes para la posición, radianes por segundo para la velocidad y radianes por segundo al cuadrado para la aceleración. Como se puede comprobar, se ha generado una aceleración negativa proporcional al pulso generado y, como consecuencia, los resultados de velocidad y de posición, siempre teniendo en cuenta que la posición es la integral de la velocidad y la velocidad la integral de la aceleración.

De esta primera simulación se pueden extraer diferentes conclusiones. La primera es que, para mantener una velocidad de giro constante, se deben equilibrar las fuerzas de los cuatro motores. Esto no implica que esta velocidad de giro sea nula, estado en el cual las coordenadas de orientación se mantienen. Lo que se provocaría sería un giro continuo del *dron*, lo que lo llevaría a una colisión segura. De esta primera conclusión se deduce un hecho no trivial. Para inducir una orientación en el cuadricóptero y después mantenerla se debe aplicar a los motores el mismo incremento en sentido contrario un poco antes de la consecución de la posición deseada. Este hecho se puede observar en la siguiente simulación realizada.

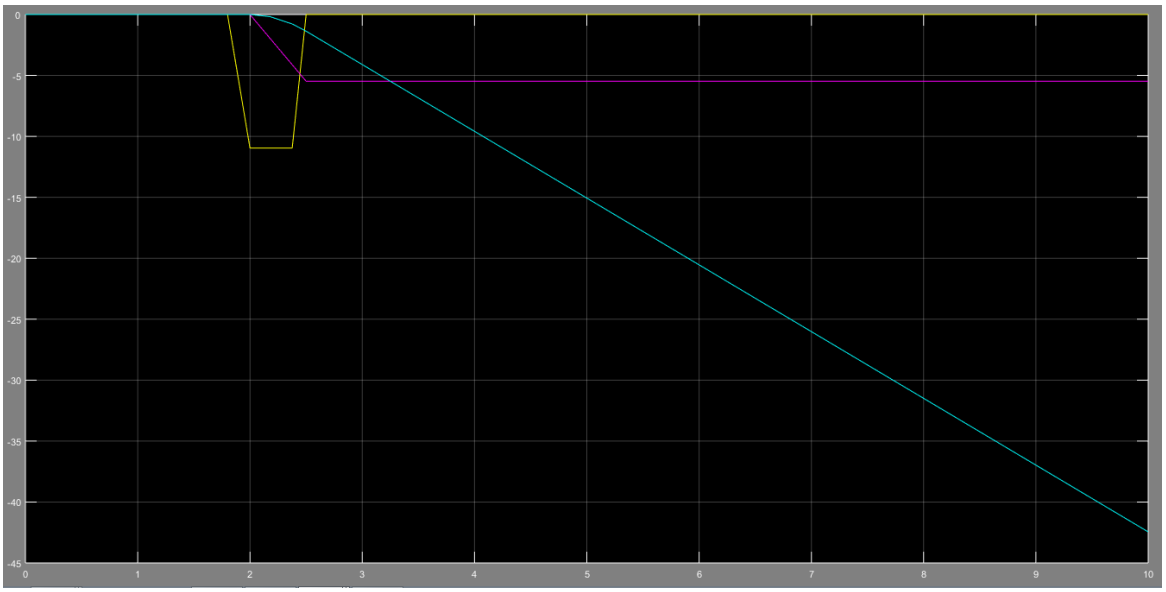

Fig. 24. Resultados correspondientes ángulo pitch (Fuente: propia)

Mantenimiento de la orientación

Para mantener la orientación del dron se debe generar, como antes se ha comentado, un incremento en el mismo motor pero en sentido contrario. En la Fig. 25 se puede observar las variaciones de la fuerza que ejerce cada motor en el *dron*. En color verde la fuerza de los motores 2 y 4, que se mantienen constantes en los valores anteriormente explicados. En color amarillo, la fuerza ejercida por el motor 1; y en color naranja la fuerza ejercida por el motor 3. Como se puede observar, los incrementos se realizan simultáneamente y de sentido contrario entre los motores 1 y 3. Esto sucede para mantener el sumatorio del momento rotacional de los motores nulos y en consecuencia para no generar un movimiento en el ángulo de yaw.

Tal como se ha deducido en la simulación anterior, también se ha generado el mismo pulso y de sentido contrario al primero en cada motor.

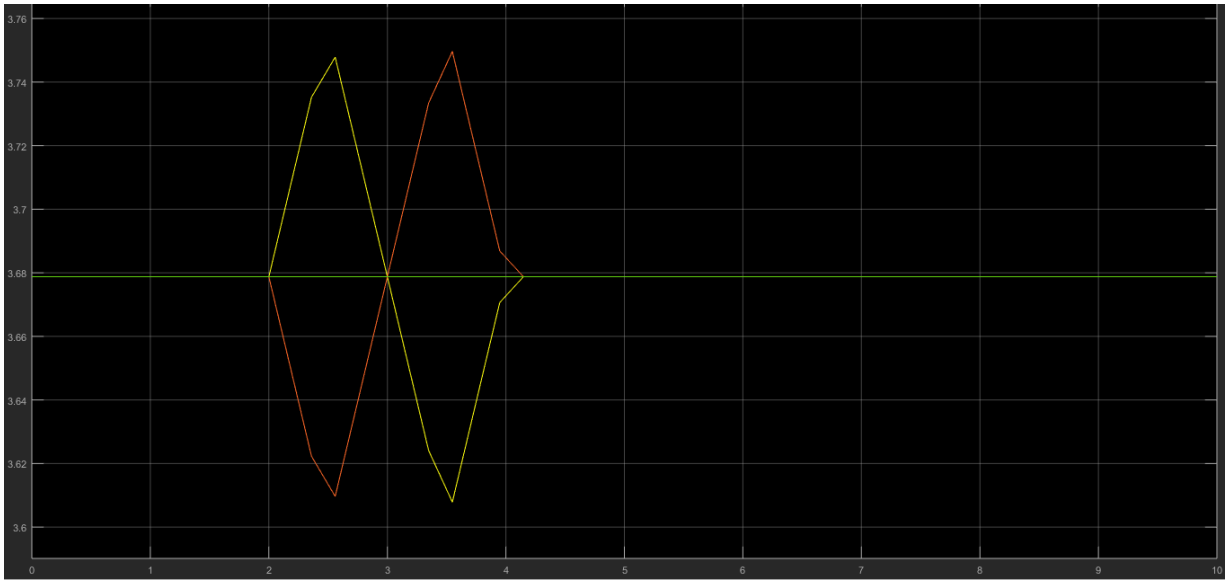

Fig. 25. Pulsos generados en los motores (Fuente: propia)

Los resultados que se obtienen, tal como cabía esperar, son los siguientes:

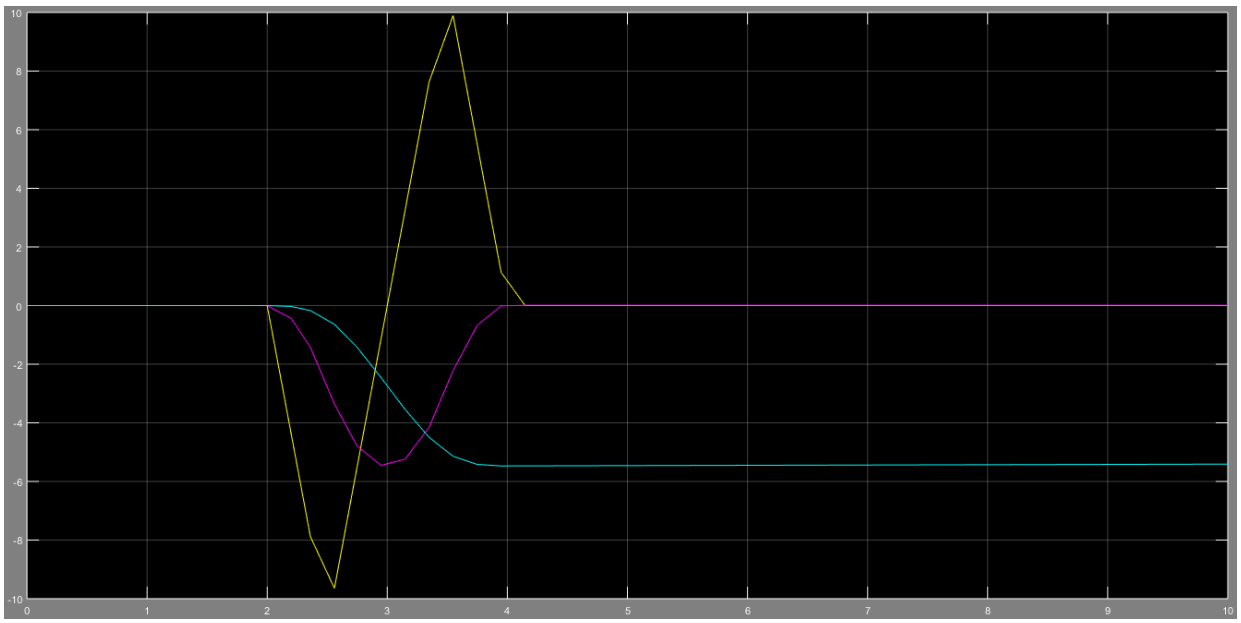

Fig. 26. Resultados de la simulación para el mantenimiento de la posición (Fuente: propia)

Como se puede observar, la aceleración angular generada toma la misma forma que los pulsos inducidos en los motores. Esta aceleración, negativa en un primer instante, genera una velocidad que se anula al generar la misma aceleración primera en sentido contrario. De esta forma, se consigue llevar a un valor nulo la velocidad, lo que hace mantener constante la orientación. En este caso esta orientación es negativa, por lo que el *dron* está inclinado hacia atrás (tomando el motor 1 el sentido de avance positivo del cuadricóptero) de manera que el *dron* se moverá longitudinalmente hacia atrás.

Simulación PLANTA-DRON

A continuación se realizaron simulaciones del sistema en lazo cerrado. Tal como se ha descrito anteriormente, esto corresponde a simular el sistema representado en la Fig. 21: las variables de entrada son los tres ángulos a controlar por el contrario de la simulación en lazo abierto de la planta en la que las variables de entrada corresponden a las cuatro fuerzas que ejercen los motores. Para ello, se opera sobre la variable error de cada ángulo, que es la diferencia entre el *setpoint* y el valor del sensor. Este valor, en el modelo matemático, corresponde a la realimentación negativa la cual toma el valor de la salida. Equivale a una realimentación unitaria negativa.

La variable error pasa entonces por los tres controladores PID los cuales tienen, cada uno de ellos, unos valores de K_P, K_I y K_D determinados. Los correctos valores de estos parámetros llevarían al sistema a ser estable y se podría ajustar diferentes características como el tiempo de establecimiento, el máximo pico de posición. Para ello se precisa de mucho tiempo, paciencia, largos cálculos de análisis que podrían formar parte de un gran trabajo de investigación.

En este proyecto, por cuestiones de tiempo, no se ha podido encontrar los valores óptimos de las constantes del controlador. Aun así, se han realizado simulaciones para comprobar el funcionamiento de la planta en lazo cerrado.

Realimentación unitaria

Para la simulación con realimentación unitaria se asignó el único valor a los PID de K_P igual a 1 y las demás constantes nulas. Se modificó la entrada del ángulo de pitch a media simulación de tipo escalón. A continuación se muestran los resultados.

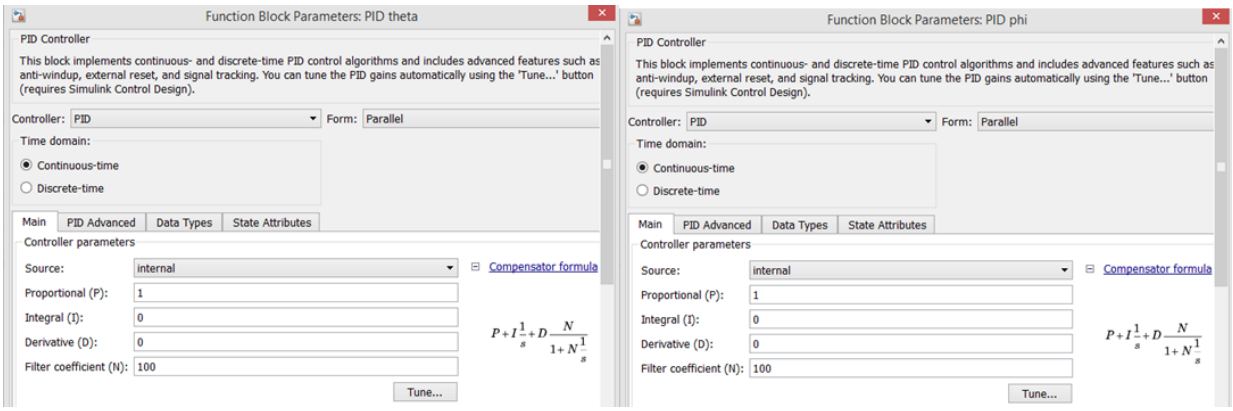

Fig. 27. Parámetros de los PID de pitch y roll. (a izquierda y derecha, respectivamente). (Fuente: propia)

Fig. 28. Variación del ángulo de pitch a los 4 segundos de simulación. (Fuente: propia)

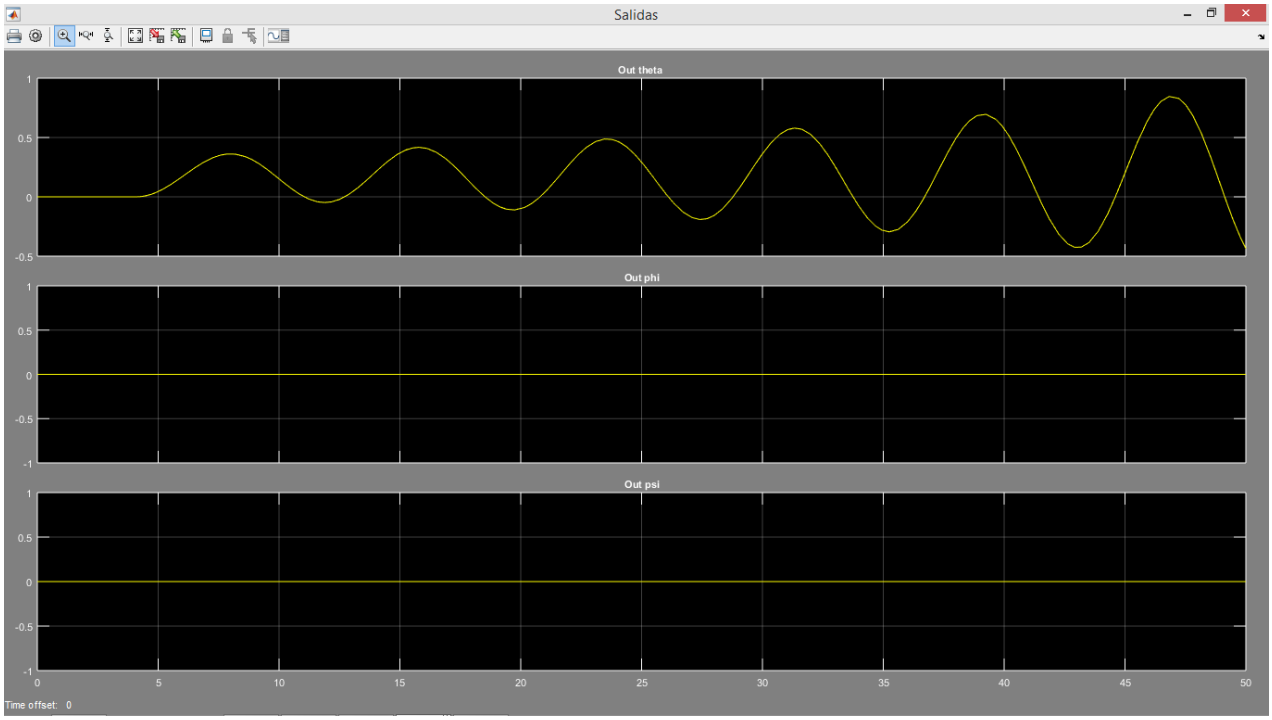

Fig. 29. Resultados de la simulación de realimentación unitaria. (Fuente: propia)

Como se puede observar en la Fig. 29, a partir de los 4 segundos de simulación el ángulo de pitch empieza a cambiar. Esto es lógico ya que el nuevo *setpoint* ha cambiado (ver Fig. 28). También se puede observar que los otros dos ángulos no han cambiado, lo que es correcto ya que lo que se deseaba, dadas las instrucciones, es que el ángulo de pitch aumentara hasta cierto valor.

El resultado preocupante y objeto de un futuro estudio es ver como la salida no converge a un valor determinado y oscila cada vez aumentando más su amplitud, con una evolución típica de un sistema inestable. El resultado de variar únicamente la K_P a 0,5 es el de reducir la amplitud de esta onda exponencial a la mitad. El resultado continúa siendo un sistema inestable.

En las siguientes simulaciones que se realizaron se aumentó los valores de K_P y se añadieron valores a K_I y K_D. Los resultados de las diferentes simulaciones tienen algo en común: el sistema es inestable, las variables de control no siguen el *setpoint* y, por lo tanto, no se puede controlar el sistema de manera tan sencilla como dando valores al azar a los parámetros de los controladores. Se precisa de tiempo y dedicación para solventar el diseño del software de control del vuelo de un *dron*.

4. Diseño de la estructura mecánica

En este proyecto se ha pasado por distintas etapas para poder llegar a lo que se ha considerado una estructura óptima. También se han realizado estudios sobre esfuerzos y de diseño para poder ensamblar los componentes básicos de los que dispone un cuadricóptero.

En primer lugar, se ha hecho un esquema que incluye todos sus componentes para después ensamblarlos.

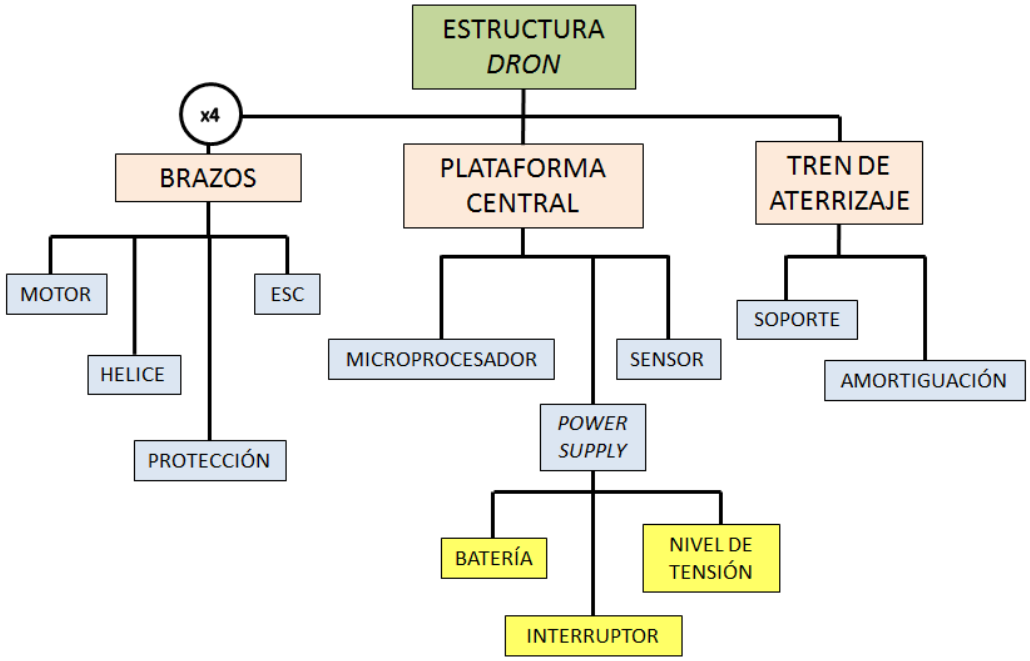

Fig. 30. Esquema de los componentes del dron (Fuente: propia).

Como se puede observar en el esquema anterior, el cuadricóptero consta de cuatro motores con cuatro hélices y sus respectivas protecciones, cuatro controladores de velocidad, un microcontrolador, un sensor, una batería con indicador de tensión, un interruptor y sistema de distribución de potencia, tren de aterrizaje y, finalmente, de un cableado para realizar sus conexiones.

4.1. Estructura de aluminio

Para la distribución de los motores se pensó en un diseño en cruz dónde la dirección del motor número 1 correspondería al avance del *dron*. El primer diseño fue de aluminio y se eligió dicho material por su capacidad de absorción de golpes y ligereza. Estaba compuesto de una plataforma central de 150x150 mm y grosor de 3 mm. Del centro salían unos brazos de perfil T, dónde estaba pensado que se dispusieran los motores, los ESC (Electronic Speed Control) y el tren de aterrizaje. Dadas las posibilidades de fabricación de que se disponía, su fabricación fue costosa a nivel de recursos. Aun así, se consiguió ensamblar las piezas y el resultado era agradable a la vista. Entonces fue cuando surgió un grave problema: por mucho que se intentara aligerar el peso con agujeros y recorte de material, al añadir los demás componentes, el conjunto del cuadricóptero iba a sobrepasar de largo el límite impuesto por los autores del proyecto de 1500 gramos para asegurar su vuelo.

Fig. 31. Primera estructura de aluminio (Fuente: propia)

Se había precisado de taladro de banco, brocas especiales para aluminio, dobladora y, en caso de necesitar recortar algún excedente, una sierra especial para metal; algo que no es fácil de conseguir si no se dispone de un taller. Ese era el caso en el que se encontraban los autores de este proyecto. Como se puede ver en la Fig. 31, se había pensado usar la superficie superior de la plataforma central para colocar el sistema de distribución de potencia, pero el exceso de tornillos para la sujeción de los elementos salientes impedía tal cosa.

Además, el aluminio es un excelente conductor. Dato muy importante a tener en cuenta si este material ha de estar en contacto con baterías, conexionado y distribución de potencia, ya que puede generar cortocircuitos y precisa de un aislamiento excelente del cual no se disponía.

Sin embargo, de este primer diseño se podría aprovechar alguna parte.

4.1.1. Estudio de esfuerzos

Para realizar el estudio de esfuerzos primeramente se realizó el diseño gráfico por ordenador mediante el programa SolidWorks. Una vez se dibujó con fidelidad cada una de las piezas, se ensamblaron mediante la opción de ensamblaje que ofrece dicho programa. Para ello, hay que insertar unas relaciones de posición relativas entre superficies, aristas o puntos de cada pieza. El resultado se puede observar en la Fig. 32.

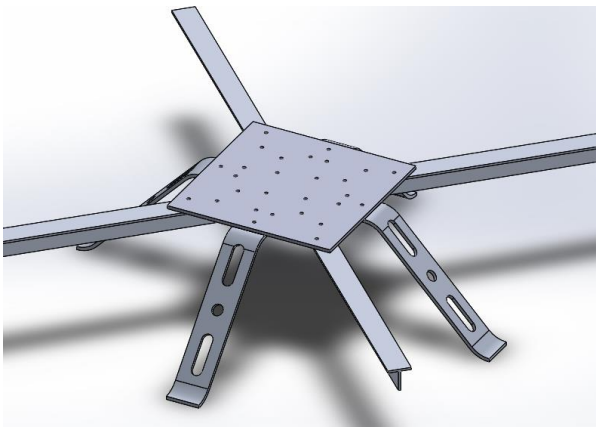

Fig. 32. Estructura dibujada con SolidWorks (Fuente: propia)

A continuación, se estudia las piezas que corresponden a las patas por separado, es decir, se aísla el problema a un solo cuerpo con unas sujeciones y unas cargas. El material de las piezas de la simulación es el mismo que se compró para la fabricación, aluminio de la aleación 1060. Sus propiedades vienen descritas en la siguiente tabla:

Material	Módulo de Young	Límite elástico	Densidad
Al1060	$E = 6{,}9 \cdot 10^{10}\ {}^{N}/_{m^2}$	$\sigma_e = 2{,}76 \cdot 10^{7}\ {}^{N}/_{m^2}$	$d = 2700\ {}^{Kg}/_{m^3}$

Tabla 3. Propiedades del aluminio 1060 (Fuente: propia)

Primero se ha realizado un estudio estático, es decir, con el *dron* en reposo en el suelo. Se ha hecho tal estudio para descartar y asegurar que una vez colocado el cuadricóptero en una superficie plana, no fuera deformando las patas para acabar en el suelo o deformándolas tanto que ya no sirvieran para su función.

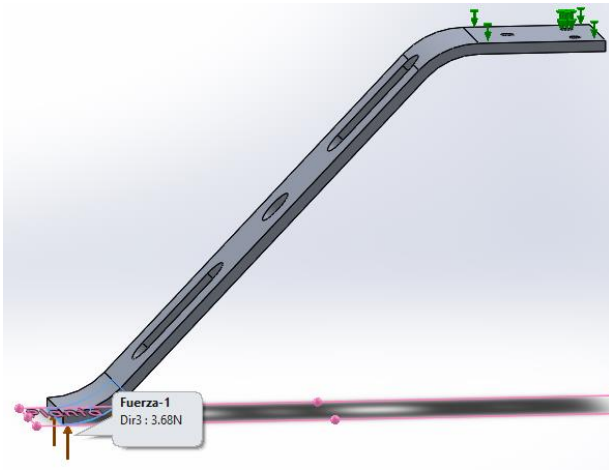

Fig. 33. Condiciones iniciales de carga en reposo (Fuente: propia)

Se planteó que la cara superior de la pata, tal como se puede observar en la Fig. 33, estuviera inmóvil, ya que es la cara que está en contacto con la plataforma central. También se planteó que en la cara inferior, la que supone estar en contacto con la superficie de reposo, recibiera una fuerza de 3,68 *N*. Este valor surge de dividir entre las 4 patas el peso total del *dron*, suponiendo que pesara los 1500 gramos máximos, y multiplicándolo por el valor de la gravedad.

Una vez asignadas las consignas en la herramienta SolidWorks SimulationXpress, se procede a la resolución y visualización de los resultados. Los resultados estudiados son la tensión de Von Mises y el desplazamiento originado al aplicar la carga. Se puede observar como la tensión máxima aparece cerca de la doblez superior y que, en este caso, no llega ni tan siquiera a la mitad del valor del límite elástico.

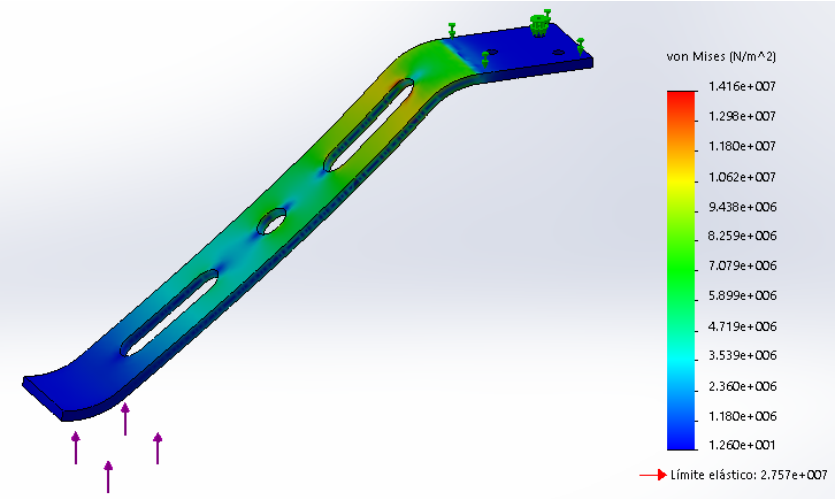

Fig. 34. Tensiones de Von Mises de la pieza (Fuente: propia)

En cuanto al desplazamiento, el máximo observado es de 0,8 mm, lo que supone un desplazamiento ínfimo y despreciable cuando el *dron* está en reposo en una superficie.

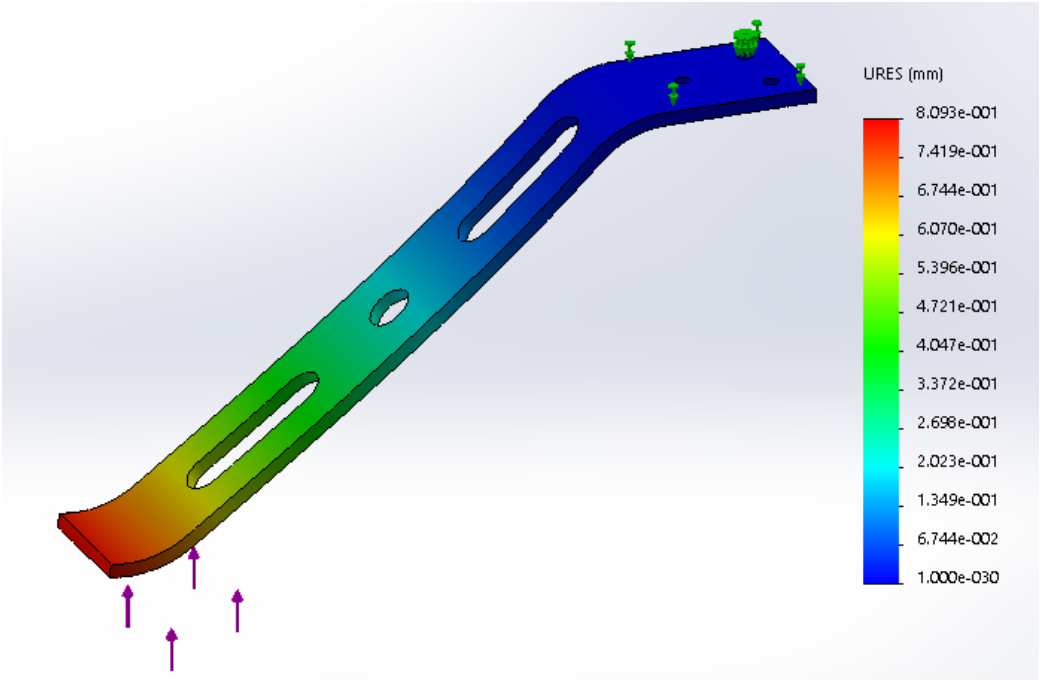

Fig. 35. Desplazamiento respecto la posicion inicial de la pieza (Fuente: propia)

A continuación se realizó un estudio que simulara una caída. Para tal cosa se imaginó una situación en la cual el *dron* simulara una caída desde 5 metros, altura considerada a la que si el dron perdiera el control y cayera, saliera casi intacto. Esta altura corresponde una velocidad de impacto con el suelo de 9,9 m/s. Si se supone que el impacto transcurre durante una décima de segundo, equivale a un pico máximo de fuerza de 149 N. Esto equivale a que cada pata reciba 37N como máximo.

$$v = \sqrt{2gh} = \sqrt{2 \cdot 9{,}81 \cdot 10} = 9{,}9 \ {}^{m}/_{s} \quad \text{(Ec. 17)}$$

$$F = \frac{mv}{t} = \frac{1{,}5 \cdot 9{,}9}{0{,}1} = 149 \ N \quad \text{(Ec. 18)}$$

$$F_i = \frac{F}{4} = 37 \ N \quad \text{(Ec. 19)}$$

Las condiciones de contorno son las mismas que para el estudio anterior. Como era de esperar, los resultados finales cambian. En cuanto se refiere a las tensiones máximas obtenidas, los valores sobrepasan el límite elástico del aluminio y, por lo tanto, habrá deformación permanente. Si además se compara el resultado obtenido en el desplazamiento se puede observar como la parte inferior, la que tiene contacto con la superficie del suelo, se desplaza un máximo de 8mm. El resultado es que la pata se deforma absorbiendo el impacto en la caída pero no llega a romperse. Este es el objetivo que debe cumplir el tren de aterrizaje del *dron*.

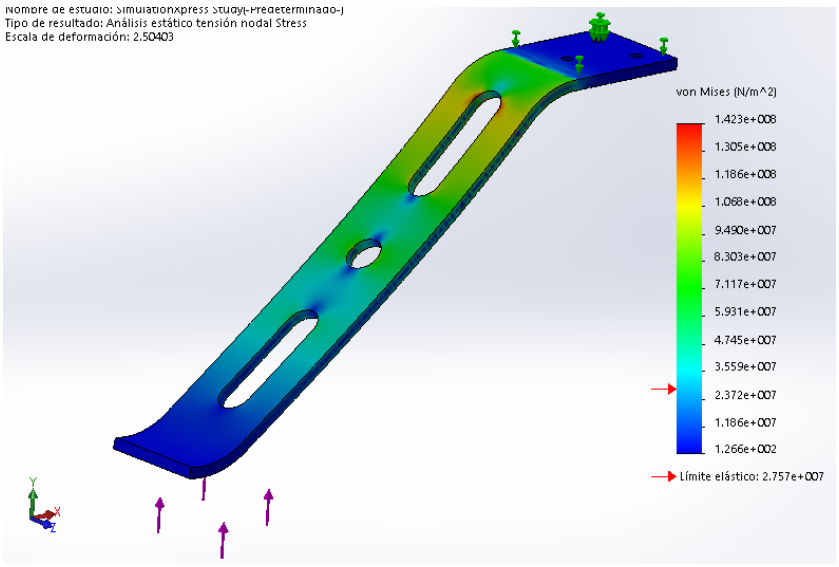

Fig. 36. Tensiones de Von Mises de la pieza (Fuente: propia)

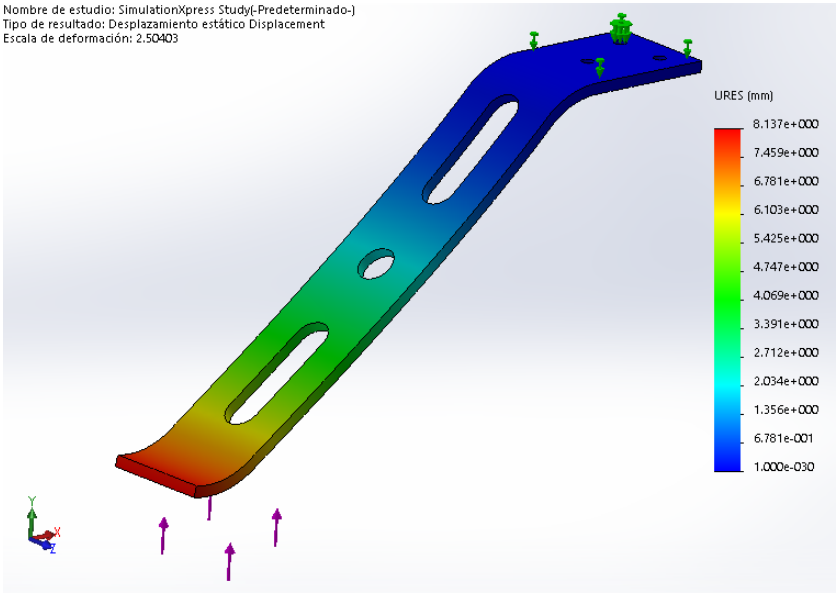

Fig. 37. Desplazamiento respecto la posición inicial de la pieza (Fuente: propia)

4.2. Estructura de metacrilato

La estructura que a continuación se describe jamás llegó a ver la luz. La razón: el elevado coste económico que supone, ya que el presupuesto realizado por una empresa especialista rondaba los 500€.

El criterio principal por el que se eligió el nuevo material fue por su ligereza y no conductividad. Además, sin saber cuánto puede costar confeccionar el diseño, se tiene completa libertad a la hora de diseñar. Para ello se usó el programa de diseño gráfico SolidWorks y el resultado obtenido fue el siguiente.

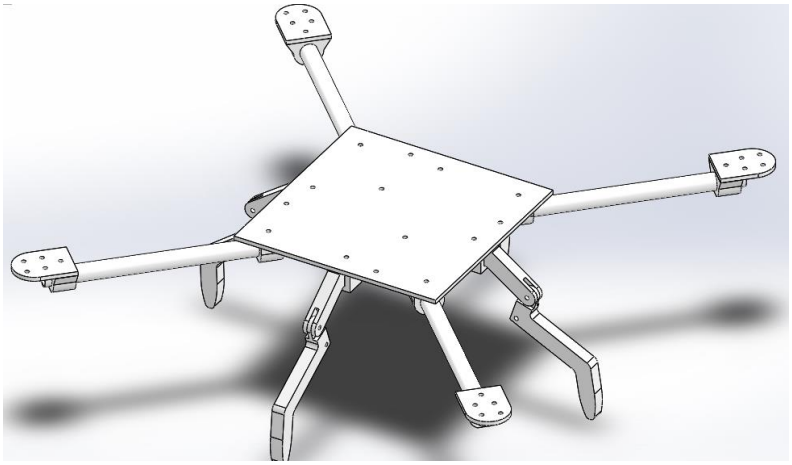

Fig. 38. Diseño de metacrilato en SolidWorks (Fuente: propia)

Con este diseño se conseguían eliminar tornillos y reducir el peso de los 900 gramos de la estructura de aluminio (sin tener en cuenta nada más que las 4 piezas correspondientes a los brazos, patas y plataforma central) a los 350 gramos. El margen es de medio quilo, lo que en estos casos, puede marcar la diferencia entre poder o no poder volar.

En la Fig. 39 se muestra con más detalle el diseño de una pata correspondiente al tren de aterrizaje. La novedosa forma está pensada para que entre el punto A y A' se dispusiera de un muelle amortiguador el cual evitaría la propagación de las tensiones por toda la estructura en los accidentes y los aterrizajes sin control automático.

En este caso, se pensó que cada una de las piezas que conformaban el *dron* fuera cortada con láser a partir de diferentes planchas con el grosor específico que ofertaba la empresa. La unión entre piezas sería con los mínimos tornillos necesarios y pegamento. Así se optimizaría el peso necesario para su fabricación.

Como ya se ha mencionado anteriormente, este diseño fue descartado por su elevado precio. Sin embargo, como ya sucediera en el diseño de la estructura de aluminio, una parte de este diseño y trabajo desarrollado ha sido útil a la hora de confeccionar la estructura final con la que el cuadricóptero volará.

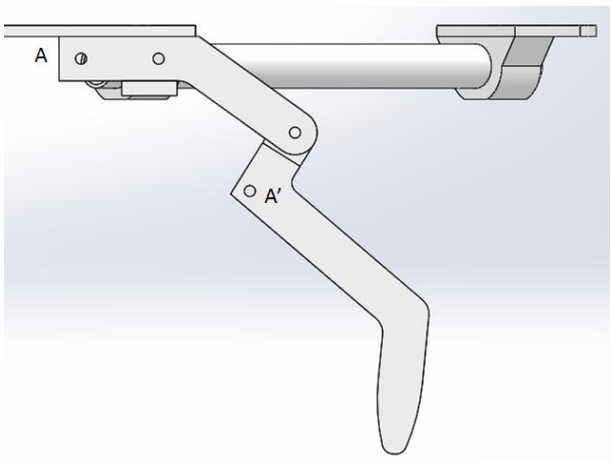

Fig. 39. Tren de aterrizaje metacrilato (Fuente: propia)

4.3. Estructura final

Básicamente, la mayor dificultad que se encuentra a la hora de construir o desarrollar cualquier proyecto es el coste económico que representa. En este proyecto se ha intentado minimizar esos costes de dos formas: siendo los autores del proyecto la mano de obra que mecanizara las piezas con la forma deseada; e intentando adquirir materiales baratos que permitieran poner en práctica la primera acción de *DIY* ("Do It Yourself"). Los materiales, aunque pudieran ser mecanizados, están pensados para soportar los esfuerzos necesarios a los que pudieran verse sometidos.

Surgió también la posibilidad de un método de fabricación muy novedoso y puntero en cuanto a tecnología se refiere: se trata de la impresión 3D. Para ello se precisa de una impresora 3D y de material de impresión. Funciona a base de calentar el extremo de un cable de plástico derritiéndolo y haciéndolo pasar, por capas, por las zonas que hay que recubrir de material para obtener, al final, la pieza deseada, tal y como se encuentra diseñada en el ordenador. Los plásticos que se usan normalmente en la impresión 3D son el PLA y ABS. Para las piezas que se precisaba construir para el desarrollo del *dron* se usó ABS, ya que es un poco más flexible que el PLA y después de un tratamiento con acetona, se consiguen piezas uniformes y brillantes.

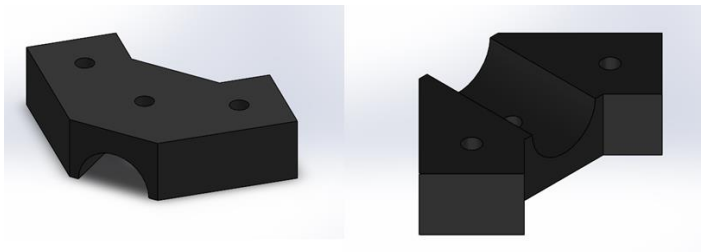

Fig. 40. Juntas de ABS dibujadas en SolidWorks (Fuente: propia)

En la siguiente tabla se enumeran los distintos problemas que se deben resolver a la hora de diseñar un *dron* (siempre teniendo en cuenta las posibilidades de cada uno) y el material elegido:

PROBLEMA	MATERIAL/PIEZA	MOTIVO
Soporte y tren de aterrizaje	Patas de aluminio, diseño de la estructura de aluminio pero con agujeros para instalar cámara, cuerdas, etc.	Son maleables, en caso de doblarse excesivamente podría recuperar su forma. Tras los estudios de esfuerzos realizados, son capaces de aguantar las caídas y los aterrizajes forzados. Con los orificios y ranuras realizadas es la parte de la estructura de aluminio que menos peso aporta.
Choque contra el suelo	Almohadillas acolchadas de plástico: material para proteger televisor dentro de caja.	Estas almohadillas absorberían parte del golpe en caso de accidente. No tienen ningún coste.

Brazos	Tubos de 12x10 de metacrilato	Tubo de 165*mm* con 1*mm* de grosor, capaza de aguantar sin problemas la flexión necesaria de los brazos entre un estado de reposo y de ascenso.
Sujeción motores	Plancha de madera	Necesidad de un material en el que se pueda trabajar con facilidad para realizar diversos agujeros con precisión.
Plataforma central	Plancha de madera	Material ligero. En esta pieza deben ensamblarse la mayoría de los componentes por lo que debe de ser fácil de taladrar y cortar.
Protecciones hélices	ABS impresora 3D	Se precisa de un material ligero y resistente. La geometría de estas piezas da poco margen a la hora de conformarla por lo que la impresión 3D resulta la mejor opción al alcance de los autores del proyecto.
Junta brazos-plataforma central	ABS impresora 3D	Se trata de 4 piezas con una geometría muy específica que, de no ser por la impresión 3D, sería muy complicado de obtener.

Tabla 4. Descripción piezas y material de fabricación (Fuente: propia)

Los planos de cada pieza se encuentran en el capítulo 2 del anexo. Una vez se está en posesión de las piezas mecanizadas se procede al ensamblaje de cada una de ellas y a la instalación de los componentes eléctricos y electrónicos.

Fig. 41. Ensamblaje de piezas y componentes (Fuente: propia)

4.3.1. *Alimentación*

El suministro de potencia cuenta con: la batería, un interruptor, cables y regletas para conectar los controladores de velocidad. La batería se ha dispuesto en la parte inferior de la plataforma central. Para ello se ha extendido un centímetro la plataforma, también con madera, para que la batería no

hiciera contacto con las patas de aluminio. En esta extensión se ha colocado velcro (de igual manera se ha colocado velcro en la batería) con el objetivo de impedir un deslizamiento en sentido longitudinal de la batería. Por si no fuera suficiente, también se ha colocado un arnés de velcro para que no cayera.

La batería tiene una junta de conexión especial de seguridad para no crear cortocircuito entre sus polos. Esto resultaría fatal en cuanto se refiere a la seguridad del cuadricóptero, ya que las baterías LiPo (más a delante se explicarán) pueden explotar y tiene una alta capacidad de descarga. De la junta del conexionado con la batería se ha instalado un interruptor que abre y cierra el suministro de potencia. Este interruptor sobresale por la parte superior de la plataforma central para tener un fácil acceso para apagar y encender el *dron*.

A continuación, de una regleta sale un cable de alimentación para la placa Arduino de 2,1mm y los cables de cobre recubierto de aislante para alimentar los ESC. Se trata de unos cables de 2mm de diámetro que han sido manipulados y dispuestos estratégicamente para alimentar los 4 controladores de velocidad. A cada lado del cuadrado que forma la plataforma central se han llevado los polos de la batería dos a dos, es decir, en los lados opuestos del cuadrado llegan los mismos polos de manera que en cada esquina se tiene un polo positivo en un lado y un polo negativo en el otro. Para poder conectar los ESC a los cables del suministro de potencia se han instalado unas conexiones de regletas. Finalmente, toda la distribución ha sido aislada con silicona para evitar cortocircuitos, para asegurar la manipulación del *dron* y para adherir el conexionado a la plataforma central inferior.

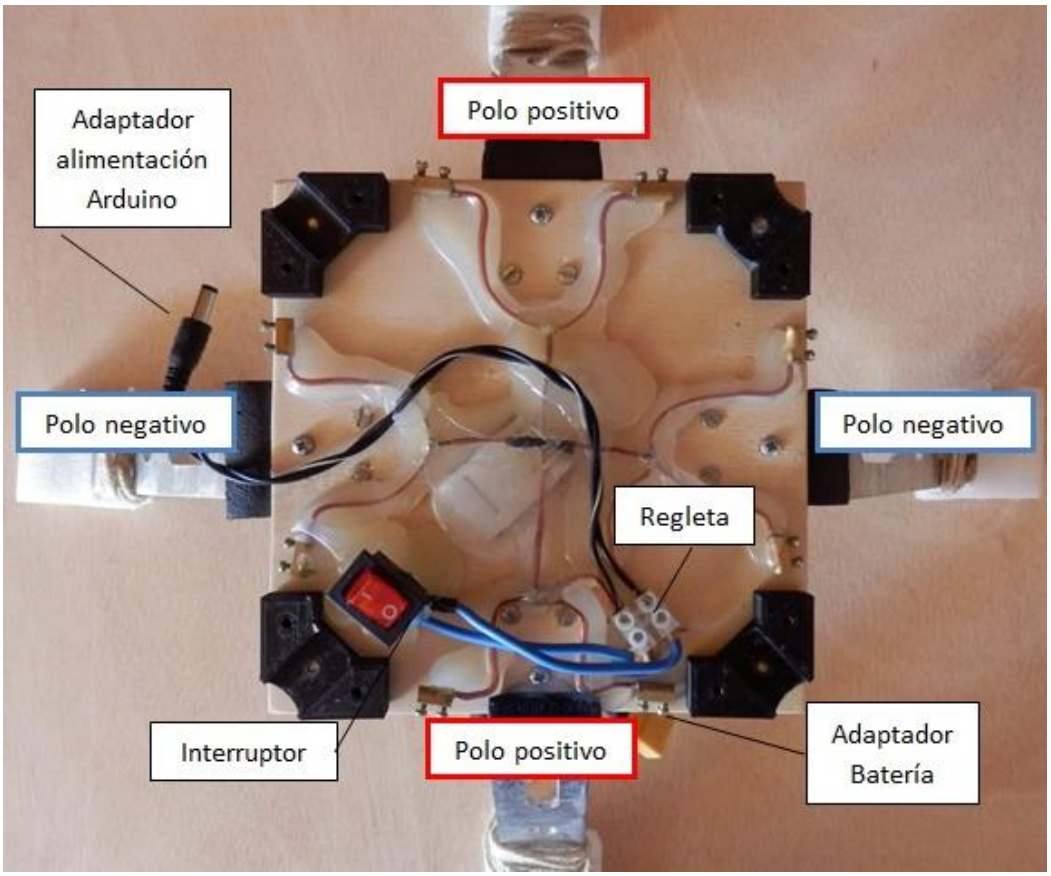

Fig. 42. Power Supply del dron (Fuente: propia)

4.3.2. Elementos electrónicos

El microcontrolador y el sensor han sido colocados en la plataforma central superior para tener un mejor acceso, aunque con ello estuvieran más expuestos. Para ello simplemente se ha servido de tornillos y arandelas.

Los ESC se encuentran en los brazos del cuadricóptero. Se trata de elementos sin enganches ni zonas por las que sea fácil introducir un cable. Carece de ranuras, por lo que se ha optado por una solución sencilla pero a la vez eficaz: cinta aislante que los envuelve entorno al brazo.

De esta manera están dispuestos cerca del suministro de potencia y del microprocesador.

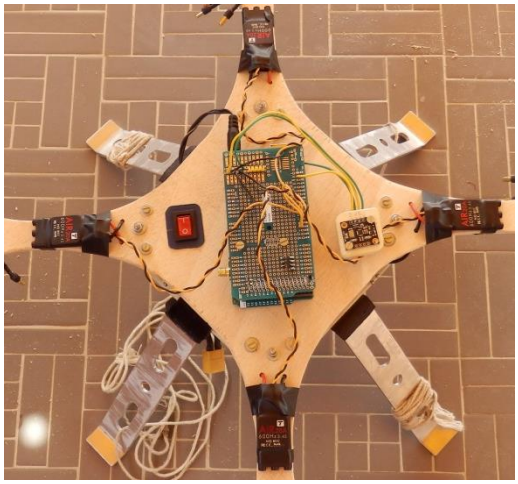

Fig. 43. Disposición componentes electrónicos (Fuente: propia)

4.3.3. Motores y protege hélices

Los motores y los protege hélices han sido instalados al final de los brazos y sobre un soporte de madera. Los motores tienen 4 agujeros en su base para introducir tornillos M3. En una primera versión se hicieron agujeros del 3 en las chapas de madera, pero el error introducido por la mano de obra no permitía una correcta sujeción. La solución fue hacer agujero del 4 y que ese pequeño juego que se dejaba permitiese el acomodamiento perfecto entre los tornillos y los motores, pasando por los las protecciones de las hélices y el soporte de madera.

4.3.4. Correcciones

Llegados a este punto, el *dron* ya está acabado a simple vista. No está listo para volar, porque falta todo el software pero sí está listo para sesiones de fotos. Y es en este punto cuando los autores de este proyecto se dieron cuenta que los motores no estaban completamente alineados, ni a la misma distancia de dónde se suponía que debía estar el centro de masas y, lo que parecía peor, estaban torcidos. De esta manera, los motores girando a la misma velocidad no ejercerían las componentes verticales de fuerza necesarias ni equitativas, además de introducir componentes horizontales que harían desestabilizar el *dron*.

Como solución al problema, se decidió cambiar las chapas individuales que tenía cada motor por un elemento único que incluyera la plataforma central superior, los brazos y los motores. De esta forma se aseguraba que los motores estuvieran a la misma distancia del centro y que no estuvieran inclinados. Aunque este nuevo elemento de madera incluyera los brazos, no se eliminan los brazos de metacrilato por la única razón de que son los responsables de añadir rigidez a este elemento tan importante. Una chapa de madera de 5mm de grosor no es capaza de aguantar la fuerza de estos motores por un lado y el peso del cuadricóptero por el otro.

Esta nueva pieza está unida a la plataforma central inferior con las juntas de ABS, que sujetan a su vez los brazos de metacrilato, y tornillos pasantes.

Fig. 44. Fabricación plataforma central y soporte motores (Fuente: propia)

Para finalizar, un elemento instalado al final e independiente de toda la estructura es una jaula de Faraday en la parte superior de la plataforma central superior. Se trata de una caja de plástico recubierto de aluminio y conectado a la tierra del *dron*, es decir, al polo negativo del *Power Supply*. Esta jaula de Faraday artesana tiene el objetivo de aislar el microcontrolador y el sensor del ruido eléctrico y de las vibraciones de las hélices. Funciona bajo el principio por el cual un campo electromagnético en el interior de un conductor en equilibrio es nulo.

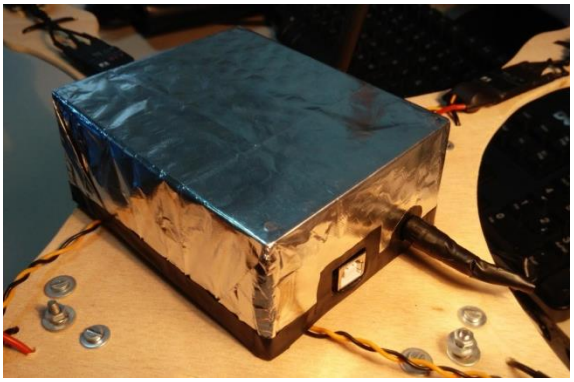

Fig. 45. Jaula de Faraday para aislar el microprocesador. (Fuente: propia)

Una vez acabado el ensamblado de la estructura se comprueba que se deja un margen de 500 gramos para la batería y otro margen de 500 gramos para tornillos, tuercas, arandelas, cableado, microprocesador, sensor y motores:

PIEZA	CANTIDAD	PESO (gramos)
Pata aluminio	4	160
Plataforma central inferior	1	38
Plataforma superior	1	79
Tubo de metacrilato	4	27
Sujeciones ABS	8	23
Protege hélices	4	180
TOTAL		507

Tabla 5. Cálculo del peso de la estructura (Fuente: propia)

5. Selección de motores

Los motores son los elementos con los que el cuadricóptero puede modificar su estado y generar un movimiento. Son los actuadores del sistema *dron* y representan la punta de lanza en la cadena de componentes que logran hacer volar al cuadricóptero.

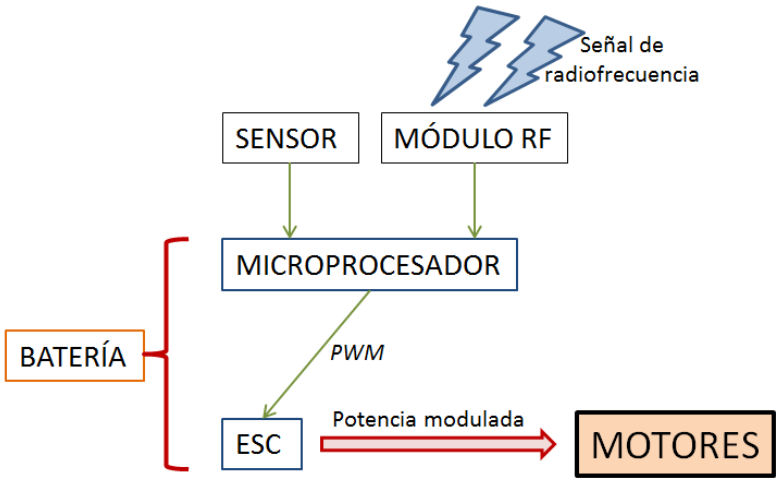

Fig. 46. Componentes del sistema dron (Fuente: propia)

Para su selección, los autores de este proyecto han sido guiados por los foros y comentarios de gente más entendida en la materia. Además, tal y como se hizo en el capítulo 2, se analizaron los modelos ya existentes para que sirviera como guía a la hora de empezar la búsqueda y la selección de algún componente.

Primeramente se había de decidir entre motores de combustión o eléctricos. En este proyecto, como se puede deducir por lo ya explicado, se decidió por los motores eléctricos. Las características por las que uno se puede decantar al usar estos motores son el bajo peso de los motores, el poco mantenimiento necesario en comparación con uno de combustión y su alta capacidad de generar potencia con baterías adecuadas que los acompañen. Los más usados en aeromodelismo son los motores sin escobillas o más conocidos por su nombre en inglés, *brushless*.

5.1. Motores *brushless*

La característica principal de los motores *brushless* es, tal como indica su nombre, la falta de escobillas en su estructura. Se usan este tipo de motores porque se precisa de un alto rendimiento de unos motores pequeños.

En un motor de corriente continua convencional, para cambiar la polaridad del campo se usan unas escobillas, o colector de delgas, que se conectan con el rotor de manera que cuando éste gira llega a un punto en el que salta de una escobilla a otra y cambia entonces de polaridad. Este rozamiento disminuye el rendimiento (muy necesario en motores pequeños), genera calor y ruido, requiere mantenimiento y puede generar un carbón que mancha el motor de polvo conductor. Conclusión: se precisa de motores sin escobillas.

Los motores *brushless* ofrecen lo anteriormente comentado a cambio de un aporte considerable de energía, ya que tienen un alto consumo. Por ello se suelen alimentar de baterías Litio-Polímero (LiPo a partir de ahora) capaces de suministrar corrientes del orden de la decena de amperios. Sin embargo, estos motores son más caros y necesitan un circuito de control de velocidad (ESCs), lo que puede duplicar el costo del conjunto.

Estos motores tienen el rotor exterior, por lo que se denominan *outrunner*. En la realización de este proyecto se han sucedido varios accidentes y posterior reparaciones. En uno de estos casos se abrió uno de los motores para ver si se podía hacer algo al respecto y, aunque no se consiguió arreglar el problema del ruido, se consiguió averiguar cómo estaban diseñados estos motores. Como ya se ha comentado, el rotor es exterior y en su interior tiene imanes permanentes. En su interior se encuentra el estator con su respectivo bobinado. Este bobinado tiene tres cables de salida, ya que estos motores funcionan con corriente alterna trifásica y es por este motivo que se necesitan los controladores de velocidad: para convertir la corriente continua en alterna y para modularla.

Fig. 47. Motor brushless outrunner de AirGear (Fuente: propia)

Los parámetros que definen sobretodo un motor *brushless* es la constante K_v. Esta constante indica la cantidad de vueltas, en revoluciones por minuto, que es capaz de dar el motor por cada voltio de corriente continua aplicado al ESC. De la velocidad de giro y según el diseño de la hélice se conseguirá una fuerza de empuje variable.

5.2. Hélices

La forma y material de las hélices determinan que fuerza de empuje y resistencia al avance se genera. En este proyecto no se ha querido entrar mucho en materia de la mecánica de fluidos y estudios de flujos entorno a cuerpos, así que básicamente la elección de las hélices es algo que iría ligado a la elección de los motores, ya que la variedad de hélices en el mercado es muy amplia. Como se puede observar en muchos foros y según comentarios de gente entendida en la materia, hay hélices que ligan mejor con según qué motores se escojan.

En un cuadricóptero dos motores giran en sentido horario y dos en sentido anti horario, por lo que la geometría de las hélices deberá ser adecuada al sentido de giro del motor en el que estén instaladas.

Sus dimensiones características vienen dadas en pulgadas y el primer valor determina la longitud de extremo a extremo. El segundo valor indica lo lejos que la hélice se movería a través del aire por cada vuelta del motor (o de la hélice). Sin embargo, este segundo valor es teórico y debe servir simplemente de guía, ya que las condiciones reales impedirán que se avance tal distancia.

Para multicópteros existe una gran variedad desde 5x3" a 9x4.5" y, en general, cuánto mayor es el aparato que se desea hacer volar, mayores son las hélices que se usan.

5.3. Elección final

El análisis y estudio para encontrar el motor y las hélices óptimos para un cuadricóptero dado podría ser objeto de un trabajo final de grado. Como no es el objeto de este proyecto se determinó un rango de motores y de hélices válidos, con el objetivo principal de ser capaces de levantar del suelo un artefacto de 1500 gramos. También, dadas las posibilidades económicas y siempre con el objetivo de coste mínimo, se miró si cabía la posibilidad de ahorrar de alguna manera. La única forma que se vio en la que se podía conseguir descuentos era en los packs de componentes, pensados para gente de espíritu *DIY*.

En el sitio web de RC Innovations [www.rc-innovations.es] se encontró un pack llamado T-Motor AIR GEAR 350 Set en el cual se incluye:

- cuatro motores *brushless outrunner* con arandela de recambio
- cuatro controladores de velocidad
- cuatro hélices

El precio total del pack es de 120€ y de esta manera se ahorra hasta 80 euros según el caso. En las especificaciones se comenta que es ideal para *drones* de 1500 gramos y está pensado para ser alimentado por una batería LiPo de entre 3 y 4 celdas (11,1 ó 14,8 volts).

Los motores tienen un K_v de 920, algo bajo según lo visto en el mercado pero, si se tiene en cuenta que los motores se pueden llegar a alimentar a 12 voltios, el máximo de revoluciones por minuto no queda demasiado corto. Admite un máximo de 18A por motor. Las hélices, tal como se ha comentado, son de 9,5 pulgadas (24,13 cm) por 4,5. En la Tabla 6 se muestran las prestaciones del pack.

Item No.	Volts (V)	Prop	Throttle	Amps (A)	Watts (W)	Thrust (g)	RPM	Efficiency (g/W)
AIR 2213 KV920	11.1	T 9545	50%	2	22.2	240	4400	10.81
			65%	3.8	42.18	386	5900	9.15
			75%	5.5	61.05	490	6900	8.03
			85%	7.2	79.92	594	7800	7.43
			100%	9.8	108.78	722	8300	6.64
	12		50%	2.3	27.6	278	4800	10.07
			65%	4.4	52.8	445	6300	8.43
			75%	6.2	74.4	568	2200	7.63
			85%	8.1	97.2	679	8100	6.99
			100%	10.9	130.8	813	8900	6.22
	14.8		50%	3.3	48.84	403	5700	8.25
			65%	6.2	91.76	636	7600	6.93
			75%	8.4	124.32	786	8600	6.32
			85%	10.7	158.36	907	9500	5.73
			100%	14.3	211.64	1084	10200	5.12

Tabla 6. Datos técnicos del pack de AIR GEAR. Fuente: www.rc-innovations.es.

5.4. Ensayos

En el capítulo 3 de este proyecto se explica el modelo dinámico de un cuadricóptero. En él aparecen constantes y coeficientes que variarán en función del tipo de motor que se use en cada modelo, lo que hará cambiar el comportamiento del *dron*. De ahí la gran importancia de este componente en el sistema cuadricóptero. Los tres parámetros a determinar son: la ganancia del motor, la constante de empuje y la constante de torsión. Para determinar estos parámetros se han realizado unas pruebas diseñadas por los mismos autores de este proyecto en las que se ha tenido que diseñar y construir los bancos de pruebas.

5.4.1. Determinar ganancia del motor

La ganancia del motor, en el caso de este proyecto, relaciona los PWM que el microcontrolador suministra a los controladores de velocidad con la fuerza de empuje que el motor es capaz de dar. Se ha supuesto una relación de linealidad entre estos dos factores, por lo que se debe medir la fuerza que realiza un motor junto con los PWM que se suministra.

Para calcular la fuerza que realizaba el motor se usó una balanza digital, encima de la cual se instaló un soporte para el motor para que éste quedase vertical. Cada motor está pensado para girar en un sentido (según el fabricante del pack adquirido) por lo que, para que la fuerza calculada no fuera vertical hacia arriba y se dejara abierta la posibilidad de que saliera volando el banco de pruebas, se instaló en el motor las hélices de los motores que giraban en sentido contrario. De esta manera se conseguía que la fuerza se realizara en contra de la balanza.

Para medir los PWM suministrados existen dos opciones: mirar con el osciloscopio la señal que se suministraba al ESC y calcular su ancho de pulso; o, con el software y las órdenes adecuadas, visualizar por el PC el valor suministrado. Los autores de este proyecto optaron por la segunda opción, y los resultados se encuentran en el capítulo 4 del anexo.

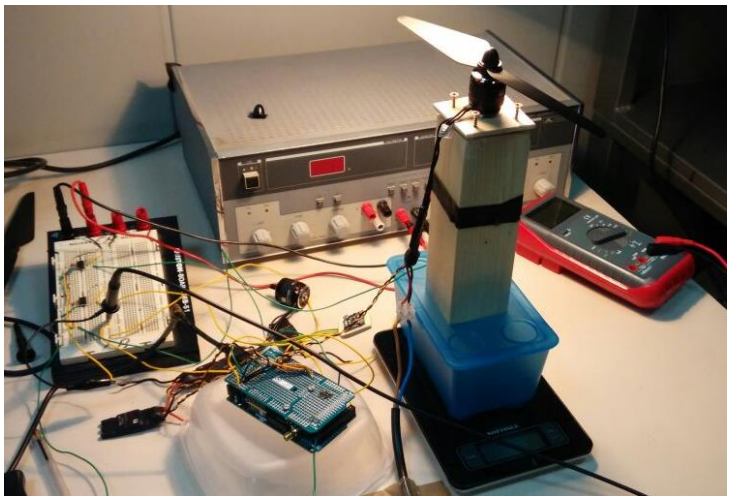

Fig. 48. Pruebas para determinar la ganancia de los motores. (Fuente: propia)

Los resultados se analizaron y según el bloque correspondiente al motor de la Fig. 18, la ecuación y gráfico que relacionan tales resultados deben tener la siguiente forma:

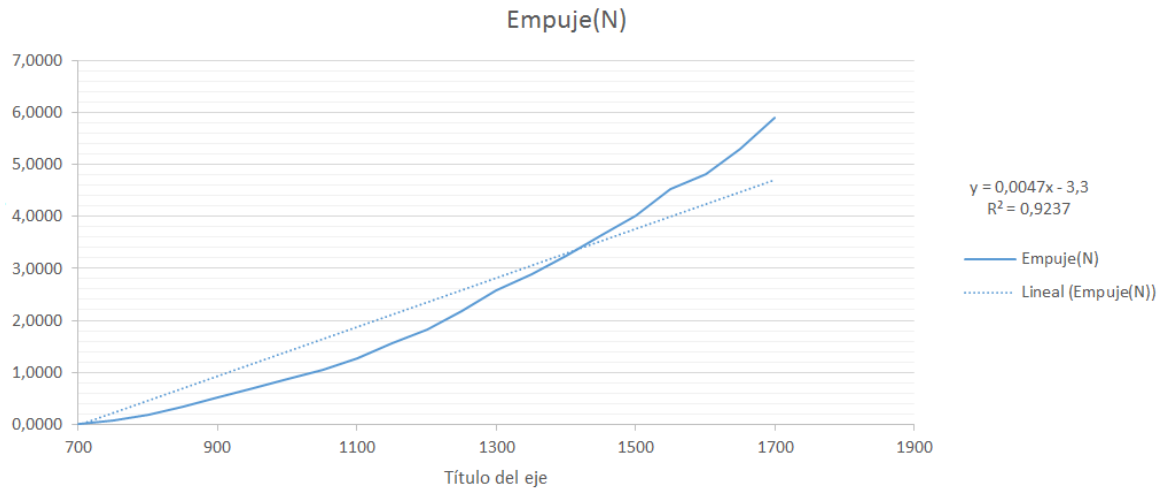

Fig. 49. Gráfica N/pwm. (Fuente: propia)

Por consiguiente, el valor de la ganancia de los motores es de 0,0047 *N/pwm*.

5.4.2. Determinar constante de empuje

La constante de empuje es aquel coeficiente que relaciona el cuadrado de la velocidad angular del motor con la fuerza que genera este.

$$T_i = K_{empuje} \cdot \omega_i^2 \qquad \text{(Ec. 20)}$$

En la gráfica de resultados del capítulo 4 del anexo se puede ver que también se han recogido los valores de la velocidad angular. Para ello, se instaló un sensor óptico debajo de las aspas del motor y de una luz. El fototransistor, cada vez que pasaba un aspa de la hélice por encima de él, cerraba un circuito y generaba una señal con una frecuencia el doble en módulo que la frecuencia de giro del motor. Dicha señal se medía por medio del osciloscopio, el cual nos daba el valor de la frecuencia de la señal. Una vez hecho los pertinentes cálculos, los resultados gráficos fueron los siguientes.

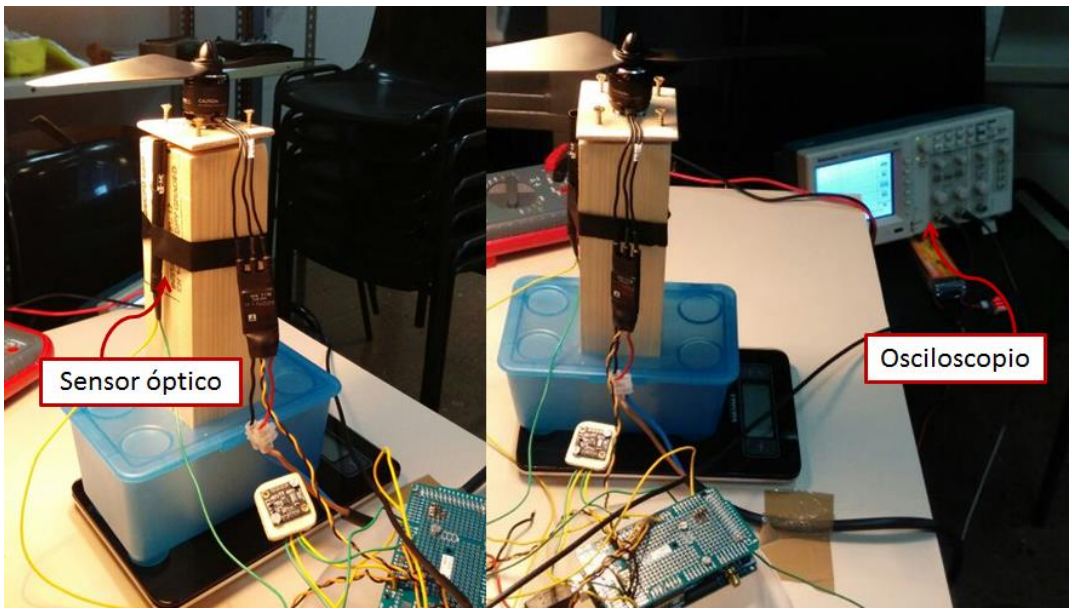

Fig. 50. Pruebas para determinar la constante de empuje. (Fuente: propia)

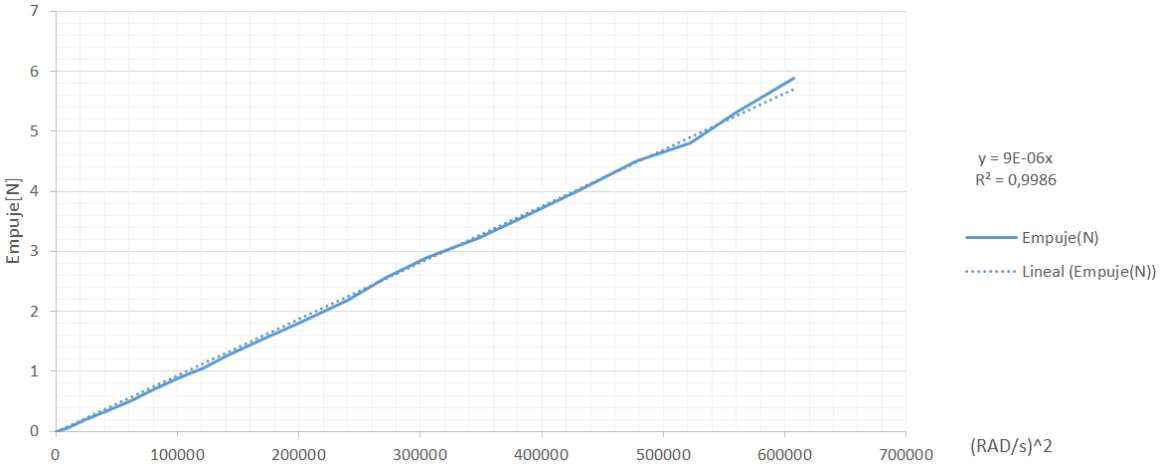

Fig. 51. Gráfica N/(rad/s)^2. (Fuente: propia)

Por consiguiente, la K_{empuje} tiene un valor de 0,000009 $N/(\frac{rad}{s})^2$.

5.4.3. Determinar constante de torsión

La constante de torsión, o K_{drag}, aparece en el momento en que un objeto gira más rápido que otro al que está unido. Es el caso de los motores y el cuadricóptero. Para calcular esta constante se tuvo que pensar y fabricar otra prueba para poder ensayar con los motores. El sistema que se pensó fue el de una estructura en forma de L con la misma longitud de los dos brazos. En un extremo de la L se colocaba el motor, girando en su sentido y con la hélice correspondiente, y en el otro extremo la estructura debía tocar de manera puntual a la balanza para calcular la fuerza que ejercía. En el centro de la L, un tornillo pasante permitía el giro de manera que ya se podía calcular el par que generaba el motor al girar.

Los resultados también han sido recogidos en la tabla del capítulo 4 del anexo. Tal como se expresa en la ecuación 21, esta constante relaciona el cuadrado de la velocidad angular con el momento de torsión.

$$\tau_{dragi} = K_{drag} \cdot \omega_i^2 \qquad \text{(Ec. 21)}$$

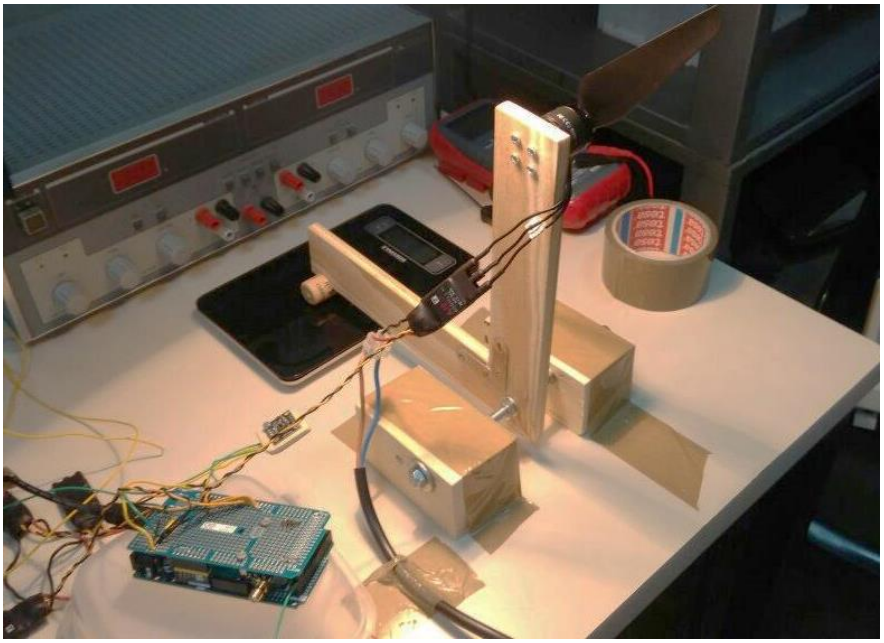

Fig. 52. Pruebas para determinar la constante de torsión. (Fuente: propia)

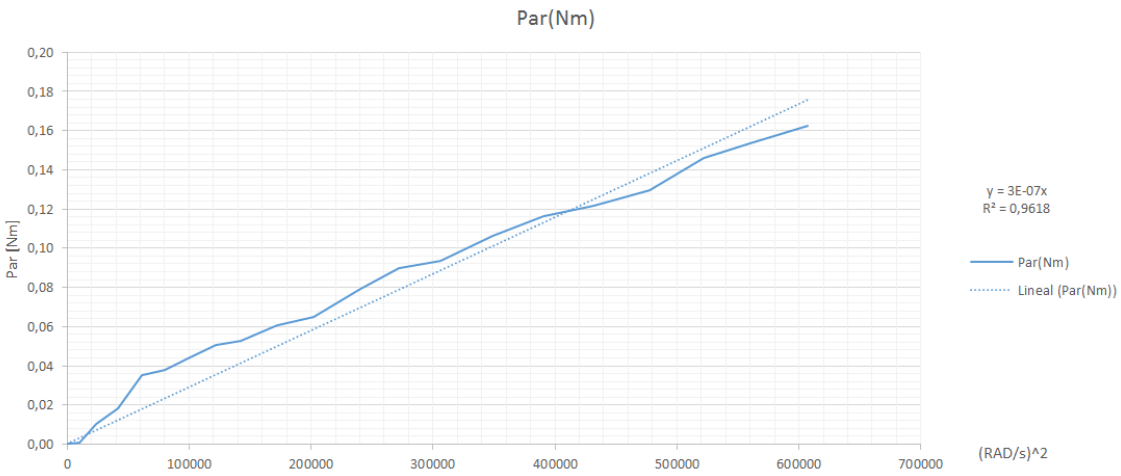

Fig. 53. Gráfica Nm/(rad/s)^2. (Fuente: propia)

Según los resultados obtenidos y tal como se puede observar en la gráfica, el valor de la K_{drag} es de 0,0000007 $Nm/(rad/s)^2$.

6. Electrónica del dron

6.1 Componentes

En el sistema electrónico de un dron típicamente se encuentran los siguientes componentes:

- El microcontrolador, que recoge la información de la unidad de medición inercial y del módulo de comunicación por radio frecuencia, la procesa y la envía en forma de señal PWM para cada uno de los actuadores.
- El módulo de comunicación por radio frecuencia, con el cual el dron es capaz de recibir las instrucciones de vuelo dictadas por el piloto a través del mando.
- La unidad de medición inercial que registra los movimientos del dron y envía la información registrada al microcontrolador.
- Los módulos que utilizan la señal PWM recibida del microcontrolador para modular el voltaje a aplicar a los motores y también actúan de convertidores estáticos de energía, ya que pasan la corriente continua que reciben de la batería a trifásica, que es la que necesitan los motores para funcionar. Estos módulos son típicamente llamados ESC (Electronic Speed Control)
- Los actuadores de tipo Brushless DC, que alimentados a través del ESC por la batería, crean un par que hace girar unas hélices y permiten volar el dron.
- La batería de tipo Li-Po (Lithium polymer battery) que alimenta el microcontrolador y los ESCs.

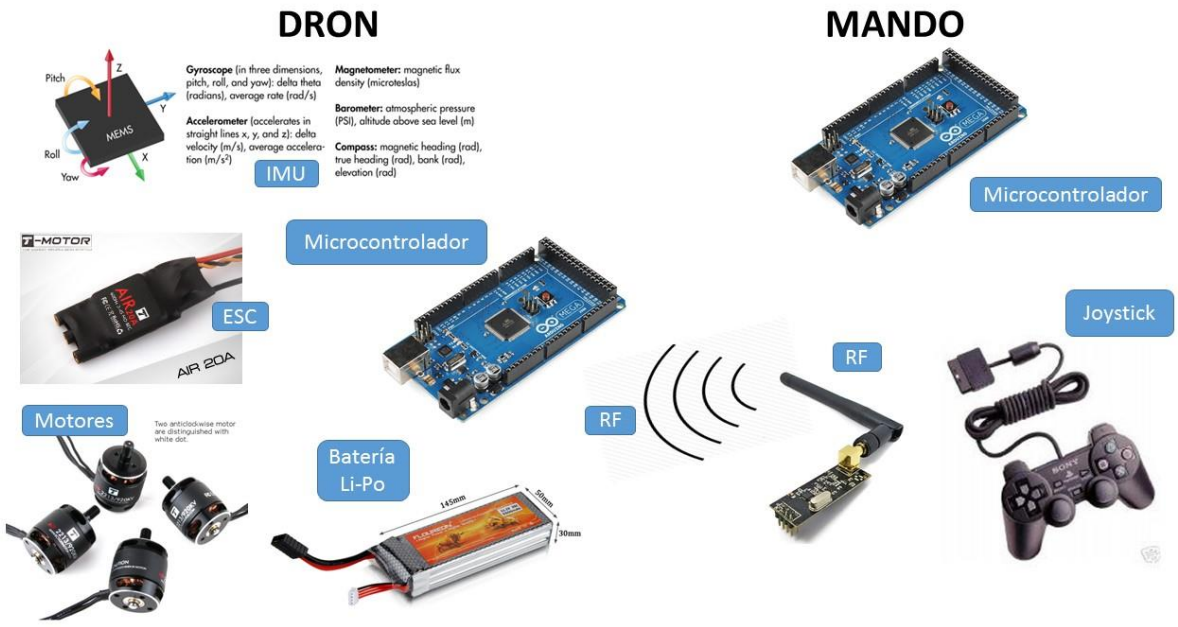

Fig. 54. Diagrama de componentes del dron (Fuente: propia)

6.1.1. Microcontrolador

El microcontrolador necesario para la realización del proyecto ha de ser uno que sea capaz de conectarse a las diferentes partes del dron para interactuar con ellas.

Para realizar esto necesitará de:

- Cuatro salidas PWM, para poder mandar a los cuatro ESC la señal para accionar a los motores.
- Protocolo SPI, para poder comunicarse con el módulo RF.
- Protocolo I2C, para poder recibir información acerca del estado de orientación del dron de la unidad de medición inercial (IMU).
- Amplio repositorio de librerías, para poder utilizar e interactuar mediante los protocolos citados anteriormente con los módulos RF e IMU, ya que no es el objetivo de este trabajo escribir cada una de las librerías necesarias.
- Interfaz de programación intuitiva y sencilla, dada la inexperiencia de los autores con microcontroladores y su programación.
- Bajo consumo y voltaje de funcionamiento compatible con las baterías a utilizar.

En el mercado se encuentran diferentes microcontroladores de arquitectura abierta. Los principales modelos y más utilizados son:

- Arduino

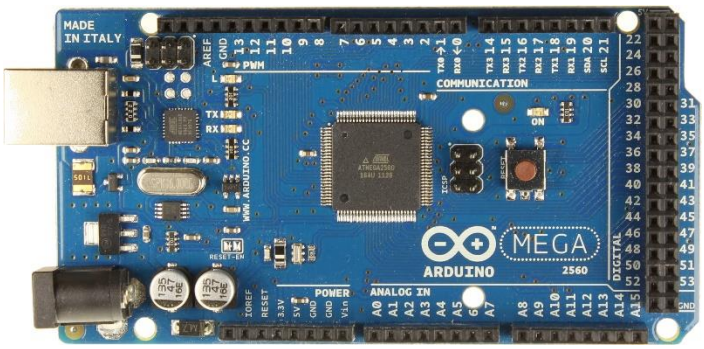

Fig. 55. Arduino Mega 2560. Fuente: www.arduino.cc.

"*Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares*."[8]

Es una plataforma de hardware libre, porque en la web están colgados los esquemáticos de todos los modelos fabricados por Arduino y pueden ser reproducidos por cualquiera que tenga unos conocimientos mínimos de electrónica.

Está basada en el microcontrolador Atmel AVR y la placa permite que los puertos de entrada y salida sean de fácil acceso.

Tiene su propio entorno de desarrollo, que permite escribir los programas a ejecutar e importar las librerías necesarias. Este entorno es continuamente actualizado para incluir mejoras y eliminar errores (bugs).

Es de fácil implantación en proyectos, dado su uso extendido por la comunidad DIY (Do It Yourself) e innumerables tutoriales para todo tipo de aplicaciones.

En concreto, el modelo a utilizar sería la placa Arduino Mega 2560, basada en el microcontrolador ATmega2560, con un voltaje de entrada de 7 a 12V y tensión interna de 5V. La frecuencia de reloj que utiliza es de 16MHz. Tiene 54 pines E/S (I/O) de los cuales 14 son salidas PWM, y 16 son entradas analógicas. Tiene 256 Kb de memoria FLASH.

- Raspberry Pi

  A diferencia de la Arduino, Raspberry Pi tiene mucho más potencial, ya que es en realidad un microcomputador en miniatura que puede correr sistemas operativos completos. Está dotada de hardware integrado para la conexión a internet, a una pantalla y también tiene una salida de audio. La velocidad del reloj es de 700MHz, con una memoria RAM de 512 MB y una memoria Flash ampliable por tarjeta SD.

Fig. 56. Raspberry Pi 2 model B. Fuente: www.adafruit.com.

- Teensy 3.1

  La plataforma Teensy es más parecida a la Arduino, ya que también permite el control del hardware (actuadores, sensores, etc.) de una manera intuitiva. Como diferencias respeto a la Arduino están las prestaciones. A continuación se muestra una tabla comparativa.

Fig. 57. Teensy 3.1. Fuente: www.pjrc.com.

	Teensy 3.1	Arduino Mega 2560
CPU	72MHz	16MHz
Flash Memory	256KB	256KB
RAM Memory	64KB	8KB
EEPROM	2KB	4KB

Tabla 7. Comparativa entre Teensy 3.1 y Arduino Mega 2560 (Fuente:propia)

De la placa Teensy 3.1 se destaca la velocidad del procesador y la memoria RAM, que son muy superiores a la Arduino Mega 2560. Las dos placas tienen a disposición varias salidas PWM necesarias para el control de los motores.

Una cosa a destacar de la placa Teensy es la cantidad de librerías que están disponibles en internet: la placa Arduino es ampliamente utilizada en la comunidad DIY, lo que hace que el número de librerías sea muy grande para todo tipo de aplicaciones; no se puede decir lo mismo para Teensy 3.1, que sí dispone de librerías pero no en tan gran variedad como Arduino.

Habiendo descrito brevemente las posibilidades de microcontroladores que parecen más adecuados se puede pasar a la elección final.

De primera mano ya se descarta la placa Raspberry Pi, ya que aunque siendo muy atractiva por el hecho de ser como un ordenador del tamaño de un teléfono móvil, necesitaría de una Arduino como complemento para poder interactuar con el hardware y eso supondría un gasto extra innecesario, aparte de ser de por sí ya más cara que las otras dos placas.

La decisión final queda entre la Arduino Mega 2560 y la Teensy 3.1, que aun siendo muy parecidas se diferencian por prestaciones, siendo la segunda mejor, y por facilidad de implementación, siendo la primera mejor. En cuanto al precio de estas, la Teensy 3.1 tiene un precio de 27 € aproximadamente y la Arduino de 15 € (modelo clonado proveniente de china). La diferencia de precio se debe sobre todo a lo explicado anteriormente respeto al gran uso que tiene la placa Arduino, lo cual ha permitido la creación de clones a muy bajo coste.

La placa elegida finalmente para su uso en el proyecto es la Arduino, remarcando como sus puntos fuertes el bajo coste, la facilidad de uso para programadores inexpertos y la disponibilidad de una amplia variedad de ejemplos y librerías en la web.

6.1.2. Módulo de comunicación por radio frecuencia

El módulo de comunicación por radio frecuencia (RF) utilizado para hacer posible la comunicación entre el mando y el dron es el nRF24l01+, basado en el circuito integrado fabricado por la empresa NORDIC y que es capaz de comunicarse en la banda de 2,4 GHz. Este módulo ha sido fabricado para aplicaciones industriales, científicas y médicas y es extremadamente útil y de fácil implantación debido a sus niveles ultra bajos de consumo, lo que permite alargar la vida de baterías de botón o de baterías AA/AAA desde meses a años.

Fig. 58. Fotografía del módulo de radiofrecuencia nRF24l01+. Fuente: www.ebay.es.

Una de las características más interesantes para el uso de este módulo es el hecho de que es un módulo de comunicación transceptor, lo que significa que puede ser utilizado para enviar y para recibir información. Gracias a su bidireccionalidad, se puede utilizar el mismo módulo tanto en el dron como en el mando.

La comunicación entre el módulo y la placa Arduino se hace a través del BUS SPI (Serial Peripheral Interface), que es un protocolo estándar de comunicación serie síncrono utilizado principalmente por microcontroladores para comunicarse con uno o más periféricos a cortas distancias o también para la comunicación entre dos microcontroladores.

En una conexión SPI siempre hay un Máster, normalmente el microcontrolador, que controla los periféricos. En esta conexión hay tres líneas que son comunes a todos los periféricos:

- MISO (Master In Slave Out). La línea del Slave para transmitir información al Master.

- MOSI (Master Out Slave In). La línea del Master para transmitir información al Slave.
- SCK (Serial Clock). Línea del reloj sincronizada con la transmisión de datos generada por el Master.

También hay otra conexión que es específica para cada periférico:

- SS (Slave Select). Esta línea se utiliza para activar la comunicación entre un periférico y el Master. Habitualmente, si SS está en un nivel bajo (LOW), el periférico se comunica con el Master y cuando está en nivel alto (HIGH), ignora el Master.

Fig. 59. Conexión SPI entre Master y Slave. Fuente: www.wikipedia.com.

Las ventajas que tiene el BUS SPI son:

- Protocolo flexible, en el que se puede tener un control absoluto sobre los bits transmitidos.
- Elección del tamaño de la trama de bits, de su significado y propósito.
- Su implementación en hardware es extremadamente simple.
- Consume poca energía, debido a que posee pocos circuitos y simples.
- Usa muchos menos terminales en cada chip/conector que un interfaz paralelo equivalente.
- Es necesaria una única señal específica para cada esclavo (señal SS). Las demás señales pueden ser compartidas.

Sus desventajas son:

- Consume más pines de cada chip que el bus I²C.
- No hay señal de reconocimiento o acuse de recibo. El servidor podría estar enviando información sin que estuviese conectado ningún cliente y no se daría cuenta de nada.

Después de una breve descripción del protocolo utilizado para la comunicación entre Arduino y nRF24L01+, se pasa a la descripción de la librería que se ha hecho servir para agilizar y hacer simple la implementación del módulo RF en el proyecto.

La librería en cuestión es la RF24, presente en el repositorio de librerías GitHub. Esta librería permite enviar y recibir mensajes de una manera fácil, y todas las operaciones necesarias para realizar estas acciones las hace la librería de forma automática, al igual que la configuración del mismo modulo RF para poder ser utilizado con el Arduino Mega.

6.1.3. Unidad de medición inercial

La unidad de medición inercial, o también abreviada como IMU del inglés Inertial Measurement Unit, es necesaria para la orientación del *dron* en el espacio. Este tipo de sensores utiliza tecnología MEMS para realizar las mediciones.

Las siglas MEMS provienen del inglés *Micro-Electro-Mechanical Systems,* es decir, sistemas mecánicos y electromecánicos hechos con técnicas de microfabricación.

En la categoría de MEMS entran dispositivos tales como microestructuras, sensores, actuadores y microelectrónica; aunque los más interesantes y los que se necesitarán para el proyecto son los microsensores, que convierten señales mecánicas en señales eléctricas.

Los sensores utilizados normalmente en una IMU son: acelerómetros, giroscopios y magnetómetros. Estos tres permiten conocer en tiempo real la orientación del objeto sobre el cual están montados.

- Acelerómetro
  Mediante el uso de sensores capacitivos se mide el diferencial de desplazamiento causado por una aceleración de una masa colocada en el sensor. Para saber la aceleración en cada uno de los ejes *X, Y, Z* se usa una masa por eje.
  Cuando la IMU se coloque sobre una superficie plana el acelerómetro medirá 0g sobre los ejes *X* e *Y* y +1g sobre el eje Z.
  A continuación se muestra un esquema representativo del acelerómetro.

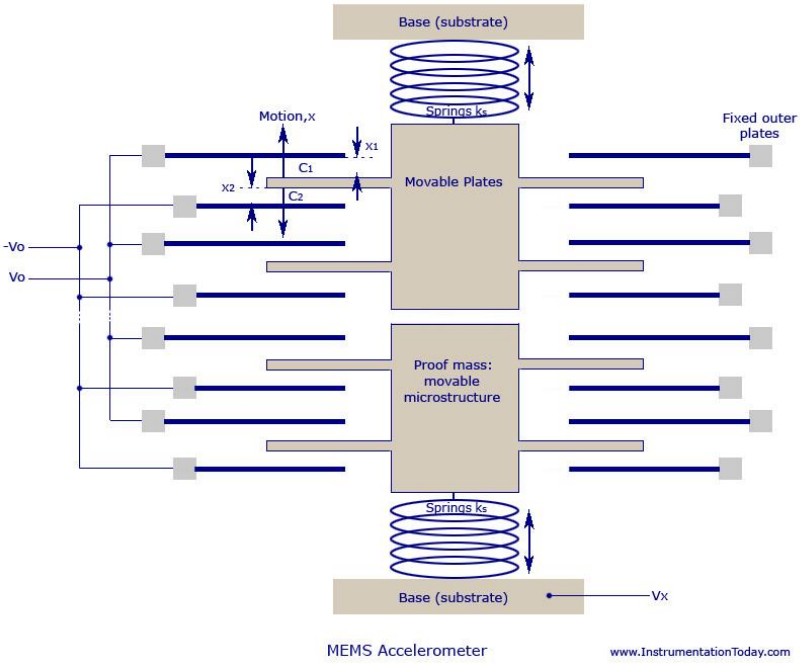

Fig. 60. Esquema estructura interna de un acelerómetro. Fuente: www.instrumentationtoday.com.

Las láminas móviles y las láminas fijas exteriores actúan como placas de un condensador. Cuando el sensor está sometido a una aceleración, la masa central se mueve y crea una capacidad entre las placas móviles y las fijas exteriores.

- Giroscopio
  El típicamente utilizado es el giroscopio MEMS de estructura vibrante, que detecta la variación de la velocidad de rotación sobre un eje. Es llamado de estructura vibrante ya que al rotar el giroscopio sobre un eje, por efecto Coriolis, se produce una vibración que es medida por unos platos capacitivos. La señal generada por los platos es tratada para que sea proporcional a la velocidad angular.

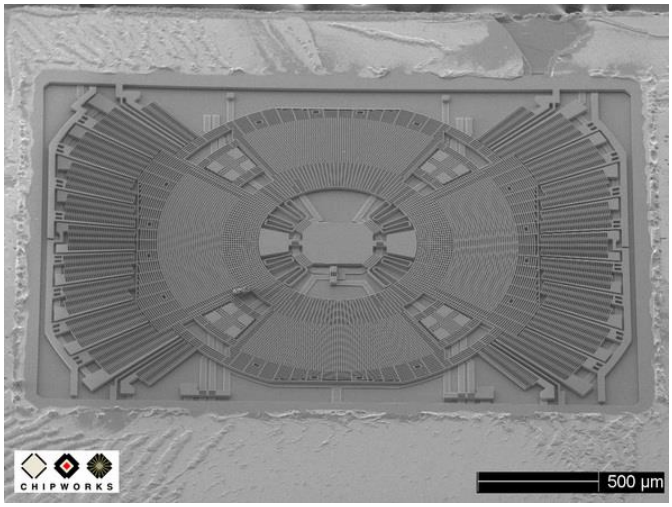

Fig. 61. Estructura interna a nivel microscópico del giroscopio. Fuente: www.ifixit.com del articulo *iPhone 4 Gyroscope Teardown* [2010].

- Magnetómetro

  Este sensor se basa en el efecto Hall, aunque no es el único. El efecto Hall en estos sensores se puede describir como la situación en la cual un semiconductor portador de corriente es colocado en un campo magnético, los portadores de carga del semiconductor experimentarán una fuerza perpendicular al campo magnético y a la corriente. Llegados al equilibrio se creará un voltaje entre las esquinas del semiconductor[9].

  El procesamiento de este voltaje permitirá conocer el valor del campo magnético aplicado. La colocación de varios sensores de este tipo de manera estratégica, representados en la Fig. 62 como 8 cuadrados, permite también conocer la dirección de las líneas de campo.

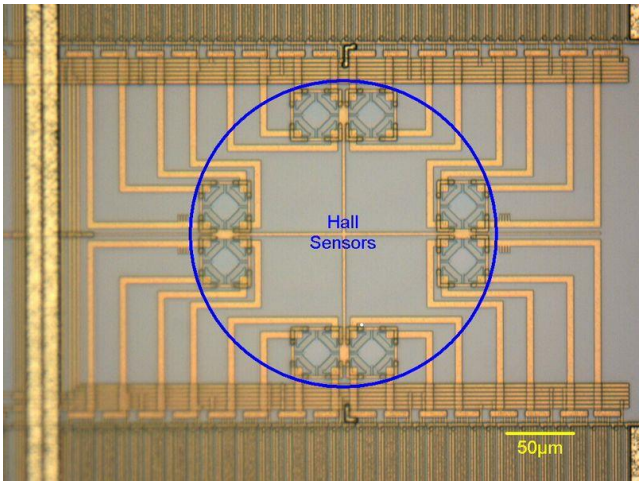

Fig. 62. Layout del sensor tipo Hall AK8973. Fuente: www.memsjournal.com.

La IMU utilizada en el proyecto está fabricada por la empresa francesa Drotek, que vende todo tipo de electrónica necesaria para la construcción de un dron. El modelo adquirido es el MPU9250, que utiliza el sensor homónimo fabricado por la empresa Invesense. Es una placa muy versátil, que dispone de *10DOF (Degrees of Freedom)* siendo estos: 3 grados de libertad del acelerómetro, 3 del giroscopio y 3 del magnetómetro. Cada tipo de sensor tiene 3 grados de libertad, ya que realiza las mediciones en cada uno de los ejes *X, Y* e *Z.* Finalmente, el último grado de libertad corresponde al barómetro

integrado, que nos permite medir la presión atmosférica para así saber la altura a la que se encuentra el sensor.

Fig. 63.Placa MPU-9250. Fuente: www.drotek.fr

La librería utilizada para comunicarse con el sensor mediante el microcontrolador Arduino es la llamada RTIMULibrary, publicada en el blog tecnológico *richards-tech.* Es una librería muy interesante, que permite leer las medidas realizadas por el sensor de manera fácil y sin complicaciones. La librería se basa en el algoritmo de fusión RTQF, creado por el mismo autor del blog, que es una versión muy simplificada de un filtro de Kalman y recolecta la mínima información imprescindible del acelerómetro, giroscopio y magnetómetro para calcular la orientación.

El algoritmo funciona utilizando las lecturas *[rad/s]* del giroscopio, junto con el tiempo entre muestras, para extrapolar de manera lineal, con respeto a la orientación previa, la orientación actual del sensor. Para evitar que los errores de predicción se acumulen, los acelerómetros y los magnetómetros proporcionan una lectura de los ángulos de pitch, roll y yaw. Los últimos dos sensores mencionados son sensores cuyas salidas contienen mucho ruido ya que están sometidos a múltiples perturbaciones. Por este motivo, se tiene que combinar las lecturas de todos los sensores para obtener resultados fiables[10].

La conexión entre Arduino y la placa del sensor se realiza mediante el bus I2C, muy utilizado en la industria para comunicar microcontroladores y sus periféricos. Utiliza dos cables para transmitir la información: uno es para la transmisión de datos y otro es para la señal de reloj.

Estos cables son el SDA y el SCL. Los dispositivos conectados al bus tienen una dirección única para cada uno. Como se aprecia en la Fig. 64, al bus I2C irán conectados dos sensores, uno es el MPU9250 que incluye acelerómetro, giroscopio y magnetómetro, y otro es el MS5611, que es un barómetro y termómetro integrado para calcular así la altura a la que se encuentra. Para elegir la dirección de cada uno de los sensores hay que cortocircuitar unos pines en la placa, que son los indicados en la placa con las líneas verde y azul.

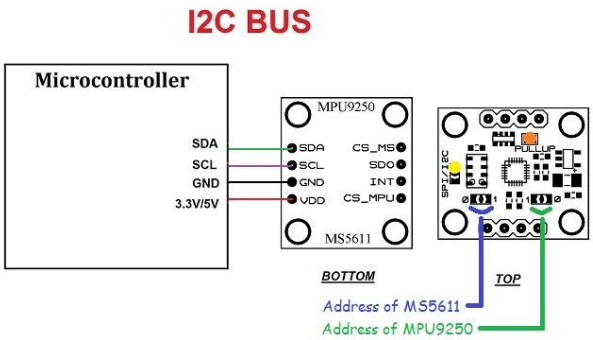

Fig. 64. Conexiones entre microcontrolador y placa sensor a través del BUS I2C. Fuente: www.drotek.fr.

6.1.4. ESC (Electronic Speed Control)

La conexión entre el microcontrolador Arduino y los motores se realiza mediante unos dispositivos de electrónica de potencia, que realizan dos funciones:

- Transformadores. Pasan la corriente continua de la batería a trifásica, que es la que necesitan los motores para funcionar.
- Reciben las señales *PWM* de los pines del microcontrolador Arduino y con ellas determinan el ratio de encendido y apagado de una red de transistores de efecto de campo MOS*FETs*, para así modular la señal trifásica destinada a los motores.

En este apartado se explicará la segunda función.

Fig. 65. AIR ESC 20A. Fuente: www.rc-innovations.es.

El cableado de los ESC consiste en la alimentación que ira conectada directamente al *Power Supply*, y otro par de cables que irán conectados a la Arduino. Por estos dos cables el microcontrolador enviara señales *PWM* necesarias para el control de los motores. El ESC elegido acepta una frecuencia de los PWM de hasta 600Hz, permite voltajes de entrada de entre 11.1 y 14.8V e intensidades de hasta 20A. Los datos de voltaje e intensidad son muy importantes, ya que en función de estos se elegirá la batería que más convenga.

Otro dato importante es que el componente no está dotado de *BEC* (circuito eliminador de batería). Este circuito actúa como un regulador de voltaje, ya que tiene como entrada el voltaje de la batería y como salida una señal de 5V que se utiliza para alimentar el microcontrolador. De allí viene el nombre de circuito eliminador de batería, ya que hace innecesario el uso de una batería destinada solo a la alimentación del microcontrolador.

El microcontrolador elegido, Arduino, acepta un valor de tensión entre 5 y 20V, siendo el rango optimo el 7-12V, lo cual nos permite alimentarlo directamente con la misma batería utilizada para alimentar los *ESC* sin necesidad de instalar un *BEC*.

El ESC, para poder funcionar correctamente y de acorde con las órdenes del microcontrolador, necesita ser configurado: es necesario establecer el valor máximo y el valor mínimo del T_{ON} de la señal PWM que se utilizará. La configuración se realiza mediante el envío de una secuencia establecida de señales que se explicara más adelante.

6.1.5. Batería Li-PO

Para alimentar todos los componentes electrónicos del dron se ha utilizado una batería de polímero de litio (Li-PO). Este tipo de batería es muy utilizada en el aeromodelismo, ya que tiene una alta densidad de energía y una alta tasa de descarga necesaria para poder alimentar los 4 motores simultáneamente.

Las baterías Li-PO se componen normalmente por diferentes celdas, en función del voltaje necesario, cada una de un voltaje nominal de 3,7V, un voltaje máximo de 4,2V y un mínimo de 3V. El valor mínimo

del voltaje es muy importante, ya que si la celda cae por debajo de este valor se daña permanentemente.

Las principales características de estas baterías que reflejan sus prestaciones son:

- El número de celdas que suele ir de 1 a 4, especificado en la parte exterior como 1S y 4S respectivamente.
- La carga eléctrica de la batería expresada en mAh.
- La tasa de carga y de descarga de las batería, se expresan como la carga eléctrica multiplicada por un factor. Por ejemplo, una tasa de descarga de 35C significa que la batería es capaz de suministrar una intensidad de hasta 35 veces su carga eléctrica. Hay que diferenciar entre la tasa de carga, que es la intensidad máxima a la que podemos cargar la batería sin dañarla, y la tasa de descarga, que como se ha mencionado anteriormente se refiere a la intensidad que la batería es capaz de subministrar.
- El peso. Esta característica es un factor muy importante para este proyecto, ya que el aumento de peso del dron se traduce en mayor consumo y por lo tanto en un tiempo de vuelo más corto.

La elección de las características de la batería ha estado influenciada por diferentes parámetros:

- Primero ha sido el voltaje que acepta el microcontrolador y el que aceptan los ESCs. Como se ha explicado anteriormente, es necesario alimentar los ESCs con un voltaje mínimo de 11.1V y máximo 14.8V y el microcontrolador a máximo 12V, ya que éste irá conectado directamente a la batería.
- Segundo está la tasa de descarga que se necesita para poder mantener el dron en vuelo a una cierta altura, pero también para poder permitir hacer correcciones en la orientación, que es cuando se producen picos de intensidad absorbida por los motores.
  Como se puede observar en la tabla de prestaciones de los motores proporcionada por el fabricante (Tabla 6) a 11.1V y a empuje máximo (*Throttle* 100%), el consumo de intensidad es de 9.8 A.
  Si se multiplica este consumo por los 4 motores de los que dispone el dron, se llega a un consumo de intensidad de máximo y dividiendo este por la carga eléctrica de la batería se encuentra la tasa de descarga mínima necesaria.
- Tercero y último, se tiene que definir la carga eléctrica necesaria para tener un tiempo de vuelo aceptable, siendo este de entre 15 y 20 minutos. Para hacer esto es necesario un cálculo aproximado del peso total del *dron*, incluyendo motores, microcontrolador, estructura y batería. Una vez calculado este peso, mediante la tabla de prestaciones de los motores se puede calcular el consumo de intensidad en estacionario del *dron*:

$$P_{DRON} = P_{ESTRUCTURA} + P_{MOTORES} + P_{BATERIA} + P_{ARDUINO} = 900g + 54g \times 4 + 300g + 40g = 1456g \quad \text{(Ec. 19)}$$

$$\frac{P_{DRON}}{Nr.\ Motores} = \frac{1456g}{4} = 364g \rightarrow I \cong 3.5A \quad \text{(Ec. 22)}$$

$$C_{MIN} = I \times Nr.Motores \times T_{VUELO} \quad \text{(Ec. 23)}$$
$$= \frac{3.5\ A}{1\ A} \times 1000\ mA \times 4\ motores \times \frac{20\ min}{60\ min} \times 1\ hora$$
$$= 4667\ mAh$$

Por lo tanto, la carga eléctrica mínima (C_{MIN}) que necesita la batería del *dron* es de aproximadamente 4700 mAh. A partir de este dato se puede calcular la tasa de descarga mínima suponiendo una batería de carga eléctrica igual a la anteriormente calculada.

$$\frac{9.8\ A \times 4\ motores}{4.7\ Ah} = 8.3 \rightarrow Tasa\ de\ descarga\ minima = 8.3C \quad \text{(Ec. 24)}$$

La tasa de descarga mínima deberá ser, por lo tanto, de 8,3 veces la carga eléctrica de la batería. A partir de ahora tanto la tasa de descarga como la de carga se expresara como *x* veces la carga eléctrica (C). Al valor calculado también hay que añadir siempre un 10% de margen de seguridad, para evitar dañar la batería.

Mirando el catálogo de la tienda online de la cual se realizará la compra, la batería que mejor cumple las especificaciones tiene las siguientes características:

Voltaje nominal [V]	11.1
Nr. Celdas	3
Carga electrica [mAh]	5500
Tasa de descarga	35C
Tasa de carga	1C
Peso[g]	465
Dimensiones [mm]	165 x 60 x 35

Tabla 8. Características Li-PO. Fuente: www.ebay.es.

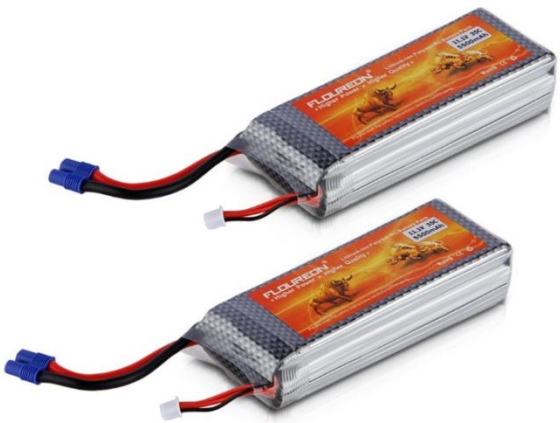

Fig. 66. Batería Li-PO Floureon 11.1V. Fuente: www.ebay.es.

Como se ha mencionado anteriormente, es muy importante que el voltaje de cada una de las celdas por la que está compuesta la batería no baje de 3V. Por eso, también se ha comprado un voltímetro que monitoriza el voltaje de las celdas y cuando una de ellas baja por debajo de un valor predefinido hace solar una alarma sonora. Dadas sus pequeñas dimensiones puede ser montado en el *dron*, sin suponer un aumento apreciable del peso.

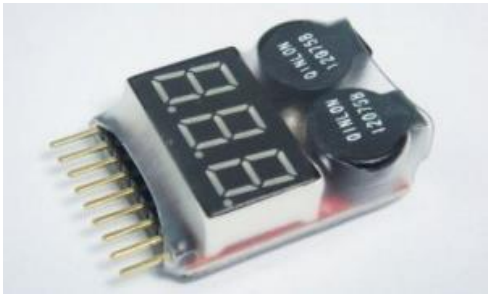

Fig. 67. Voltímetro. Fuente: www.rc-innovations.es.

6.2. Montaje

Una vez vistos los componentes electrónicos principales de los que está compuesto el dron se pasa a explicar el montaje de cada uno de ellos.

6.2.1 Mando de radiocontrol

El mando de radiocontrol es una parte muy importante de sistema, ya que permite que el piloto pueda controlar su *dron*. En la mayoría de proyectos de este tipo, cuadricópteros construidos por gente *amateur,* la opción más fácil es comprar un mando fabricado por alguna empresa, como por ejemplo Futaba, que es una de las más conocidas y de más renombre. Este mando ya viene casi totalmente configurado y solo necesita de un receptor instalado en el dron para recibir las órdenes dictadas por el piloto.

Fig. 68. Mando de Radiocontrol Futaba de 6 canales. Fuente: www.rc-innovations.es.

La opción de comprar uno de estos mandos es muy buena y sensata, ya que permite centrarse más en la construcción del dron en sí y además son muy fiables. El problema es que su precio es muy elevado, ya que va de 100€ hasta casi los 500€. Este rango de precios es inaceptable para la compra del mando y por lo tanto se ha decidido fabricar uno basado en algunos de los componentes vistos anteriormente y un joystick. Esta solución pretende así ser más barata y además cumplir con uno de los principales objetivos de la realización de este proyecto, que es el de mejorar las habilidades de construcción y de adaptación a los medios y posibilidades disponibles de los autores.

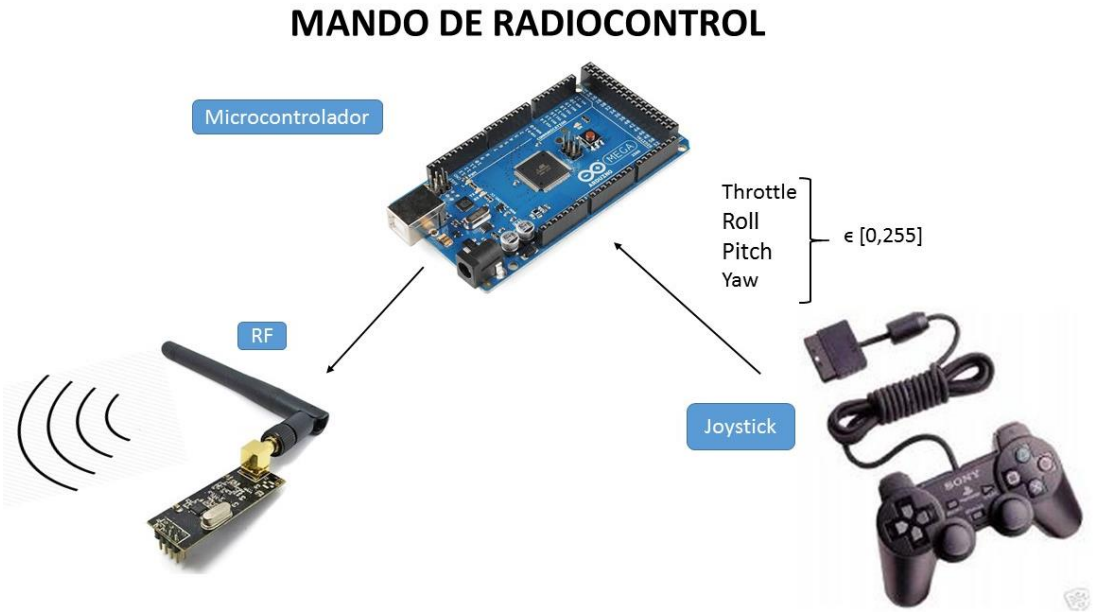

Fig. 69. Diagrama de componentes del mando de radiocontrol (Fuente: propia)

Como se puede ver en el diagrama anterior, el mando de radiocontrol estará formado por un microcontrolador Arduino, el mismo que se ha explicado anteriormente, que por una parte recibirá

señales provenientes del joystick, siendo estas las de *throttle*, *roll*, *pitch* y *yaw*, las cuales tendrán un valor entre 0 y 255. Después de recibirlas, las mandará empaquetadas trámite la librería *RF24* al módulo RF. Finalmente, este las enviará al módulo RF montado en el dron para su posterior procesamiento.

La conexión del módulo RF a la placa Arduino se ha hecho por medio de un *protoshield* específico, que permite montar y soldar componentes y tener acceso a los pines.

A continuación se puede ver el esquemático del nRF24L01+ y la tabla donde se indican cómo se tiene que conectar cada uno de los pines.

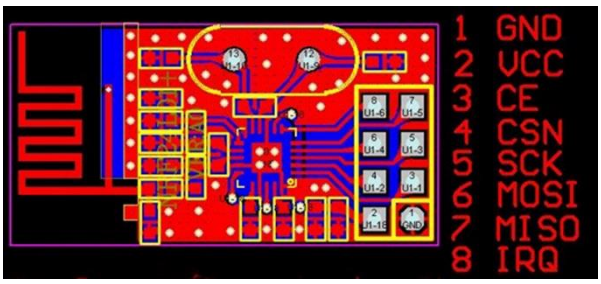

Fig. 70. Esquemático del módulo nRF24L01+. Fuente: www.arduino-info.wikispaces.com.

Señal	Modulo RF	Arduino Mega 2560 Pin
GND	1	GND
VCC	2	3.3V
CE	3	9
CSN	4	53
SCK	5	52
MOSI	6	51
MISO	7	50
IRQ	8	-

Tabla 9. Conexión entre modulo RF y Arduino. Fuente: www.arduino-info.wikispaces.com.

En la siguiente foto se muestra el cableado necesario para la instalación del módulo RF en la *protoshield*.

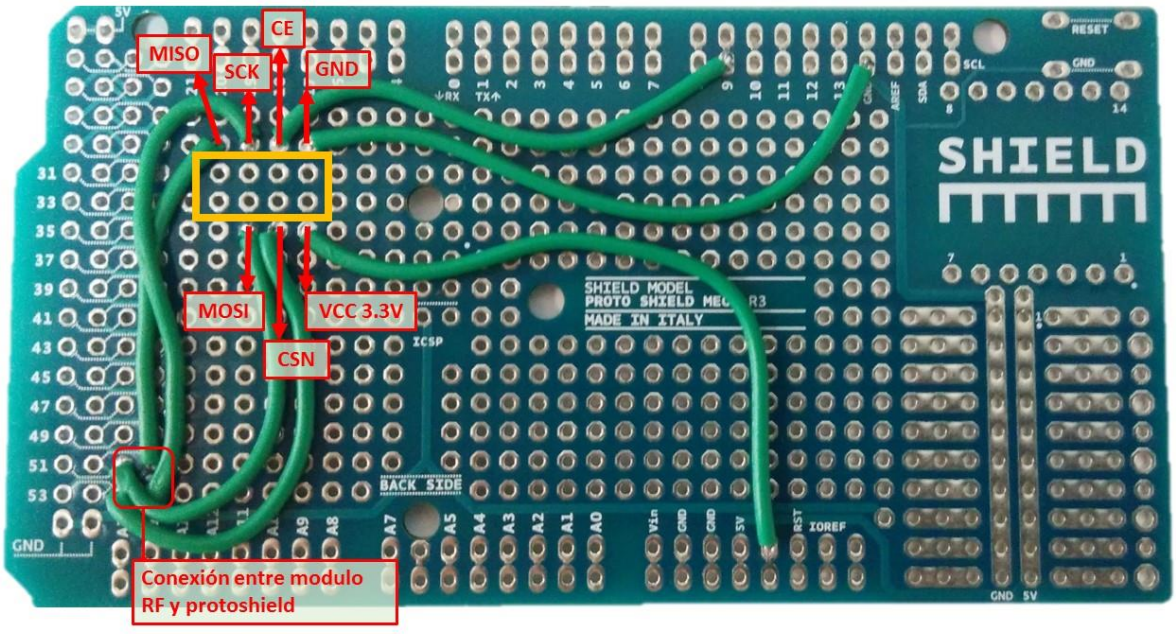

Fig. 71. Cara interior de la protoshield con el cableado soldado (Fuente:propia)

Una vez soldados los cables, se coloca el modulo RF en la placa y se suelda utilizando los agujeros que se encuentran en el recuadro naranja de la Fig. 71.

Como viene remarcado en las especificaciones, la alimentación del módulo se realiza por los pines de alimentación del Arduino de 3.3V y no por los de 5V, ya que causaría la destrucción del componente. Otro tema a tener en cuenta es que los pines del Arduino en ocasiones no son capaces de subministrar los picos de intensidad demandados por el modulo RF. Por este motivo, entre el pin de VCC y GND del módulo es necesario poner un condensador de 10 µF, que se verá instalado más adelante en la descripción del montaje. Otra función que realiza el condensador es filtrar el ruido producido por la alimentación de la placa Arduino.

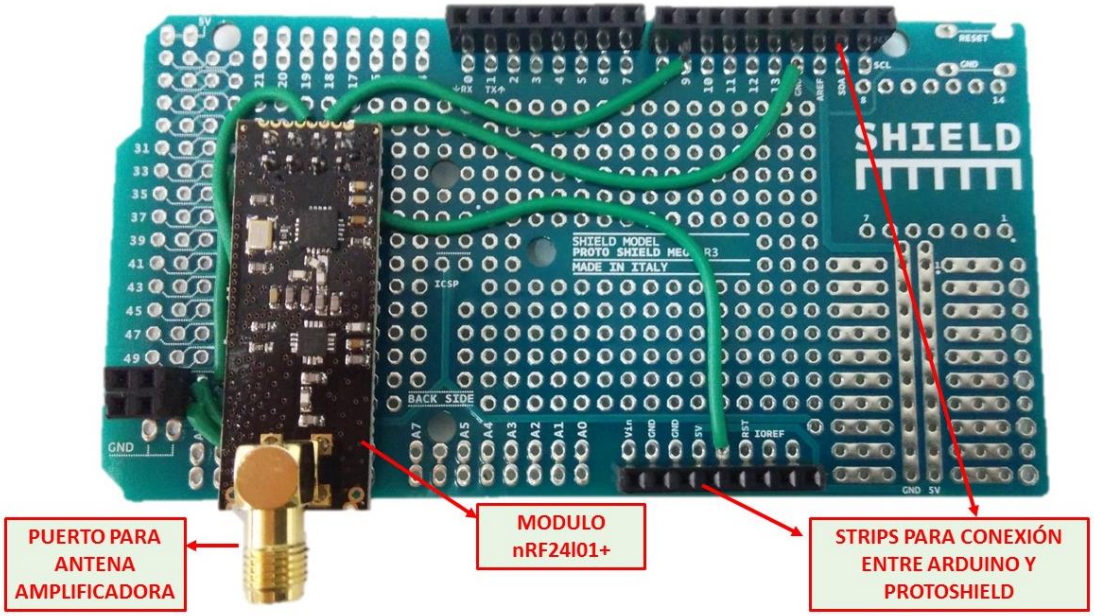

Fig. 72. Vista interior del protoshield con cableado, modulo RF y strips soldados (Fuente: propia)

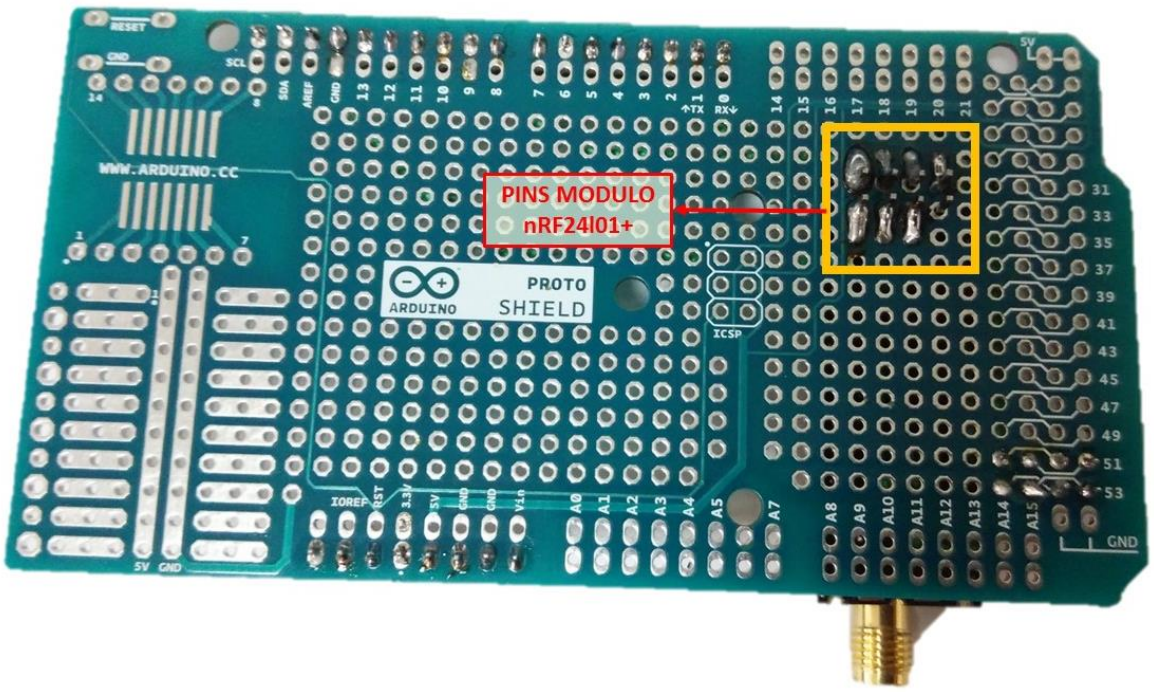

Fig. 73. Vista exterior de la protoshield (Fuente: propia)

Una vez instalado el modulo RF en la placa se pasa a hacer la conexión del mando PS2.

Para que la comunicación entre mando y Arduino sea posible, se ha utilizado una librería que mediante una serie de simples comandos permite leer las lecturas de cada uno de los botones y palancas analógicas presentes. La librería en cuestión es la *Arduino-PS2X*, publicada en el blog llamado "The Mind of Bill Porter". [11]

Para la conexión física se ha utilizado el esquemático de la Fig. 74. Como se puede ver es necesario utilizar un conversor lógico de las señales de 3,3V a 5V, ya que el Arduino trabaja con señales digitales a 5V, mientras que el mando PS2 trabaja a 3,3V.

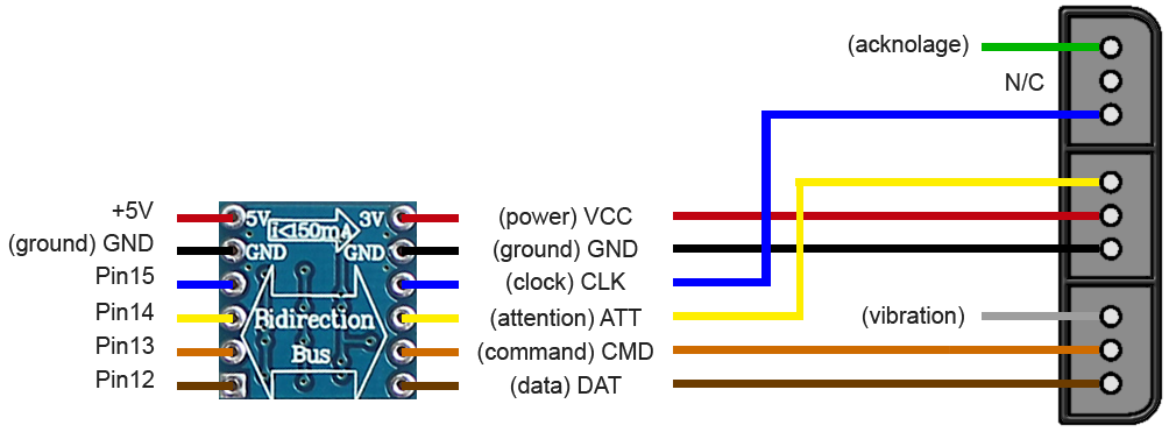

Fig. 74. Esquemático de conexión entre mando PS2 y Arduino. Fuente: www.luisllamas.es.

A continuación se muestran los dos lados del *protoshield* después de realizar las soldaduras necesarias.

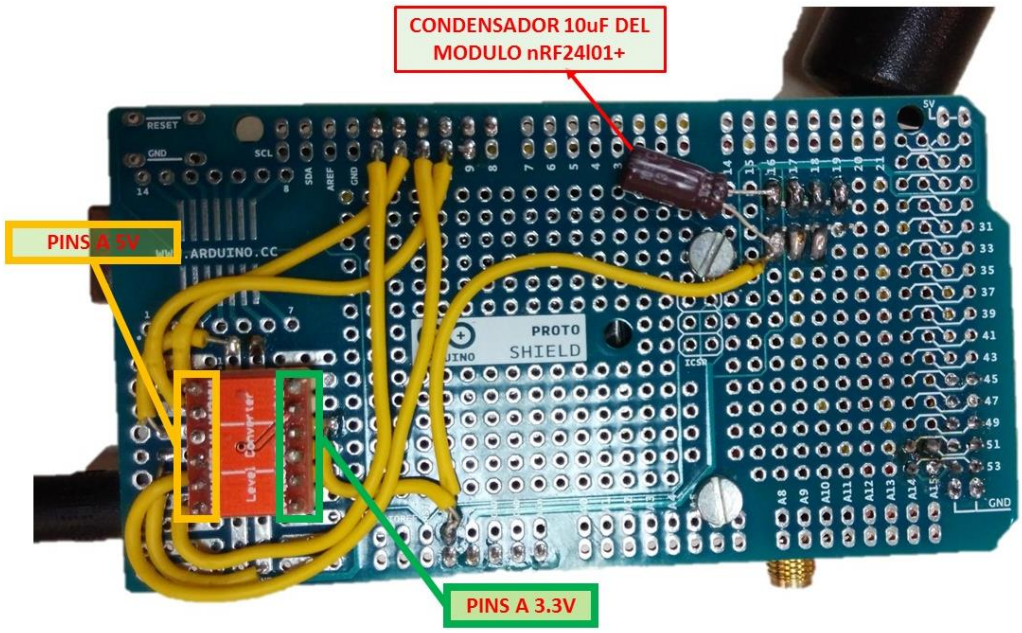

Fig. 75. Vista exterior del protoshield con el conversor lógico y los cables soldados (Fuente: propia)

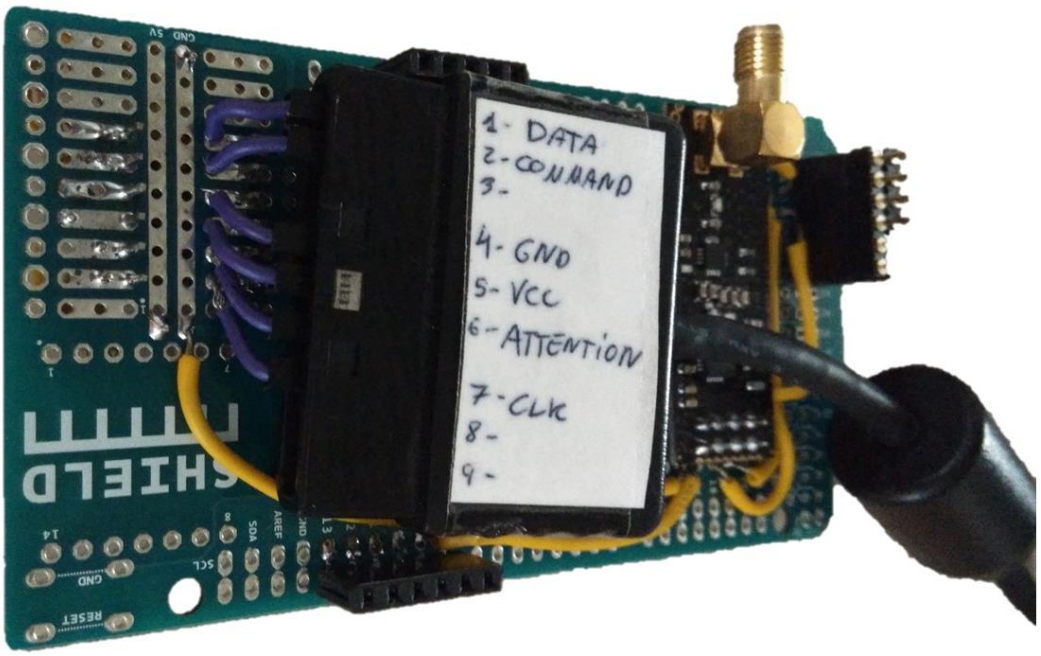

Fig. 76. Vista interior del protoshield con el conector del mando conectado (Fuente: propia)

Como se aprecia en la Fig. 77, se ha dejado el conector del mando de la PS2 intacto y se han utilizado los pines ya presentes. Para conectar estos pines con la *protoshield* se han cortado unos cables *jumpers* con conector hembra (en la Fig. 76 los cables violeta) y se han insertado en los pines; el otro lado del jumper se ha soldado directamente a la *protoshield*. De esta manera es posible extraer el conector del mando sin tener que desoldar cables, así en caso de que el mando o la placa Arduino se estropee, la substitución de uno u otro no es aparatosa.

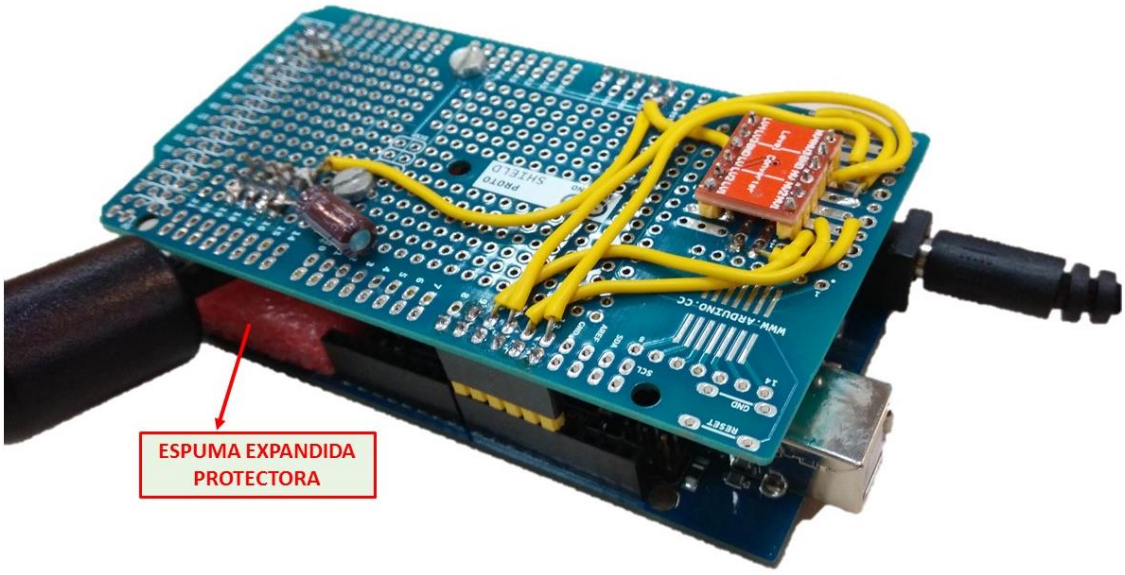

Fig. 77. Protoshield y Arduino montadas juntas (Fuente: propia)

En la Fig. 77 se muestra el resultado final, una vez montada la *protoshield* con el Arduino. Las conexiones y los componentes más frágiles quedan en el interior del conjunto para mejorar su protección. También se ha añadido un trozo de espuma expandida entre el conector del mando y el

Arduino para evitar que este choque con algún componente de la placa y lo destruya y también para minimizar la posibilidad de moverse del conector.

La alimentación del conjunto se realiza mediante 6 baterías AA recargables Ni-MH que proporcionan cada una un voltaje de 1,2V y tienen una carga de 2000mAh. El conjunto de baterías tiene así un voltaje de 7,2V, suficiente para alimentar el Arduino.

El consumo del Arduino junto con el mando *DualShock2* es algo superior a los 200mAh y siempre inferior a los 500mAh por lo que, en teoría, estas baterías recargables deberían estar operativas para usar el mando de control entre 4 y 10 horas.

Fig. 78. Mando PS2, Arduino Mega 2560 y porta baterías (Fuente: propia)

Fig. 79. Usuario utilizando el mando (Fuente: propia)

En la Fig. 78 se puede ver el mando completo: por una parte el mando de la PS2 que utilizara el usuario, la placa Arduino y el porta baterías, con un interruptor montado para apagar y encender fácilmente. En la Fig. 79 se muestra una posible manera de utilizar el mando por un posible usuario, sujetando en la mano solo el mando de la PS2 y en el bolsillo guardando el Arduino y las baterías.

El mando de radiocontrol construido en este proyecto se usa para controlar un dron, pero en realidad se podría utilizar para cualquier tipo de proyecto que necesitara de un mando, como por ejemplo un coche o un brazo mecánico.

6.2.1.1 Dificultades encontradas

La solución descrita anteriormente es la solución final, pero desafortunadamente no se ha llegado a ella directamente sino que durante el camino se han encontrado ciertas dificultades:

1. Mando PS2

   Al principio del proyecto se probaron diferentes mandos antes del llegar al utilizado.

   El primero fue un mando de la PS2 pero no el original, sino un clon chino que causo muchos problemas de compatibilidad entre Arduino y mando. La librería utilizada en las instrucciones de uso no se especifica muy bien si es posible utilizar mandos no originales de la PS2, siendo estos una opción más asequible respeto al original. Por este motivo se intentó utilizar el clon

chino, pero sin ningún resultado positivo: el mando no reaccionaba y el Arduino daba continuamente mensajes de error.
Otro intento se realizó con el mando de la PS1, pero tampoco se pudo realizar la conexión.
Finalmente, después de investigar en diferentes fórums online se encontraron los motivos por los cuales no se establecía la conexión:

- Las señales lógicas con las cuales funciona el Arduino tienen como voltajes 0V (OFF) y 5V (ON), mientras que el mando de la PS2 está configurado para trabajar con señales lógicas de 3.3V. Esta diferencia hace que el Arduino no pueda reconocer las señales enviadas por el mando ya que no puede diferenciar entre 0s y 1s.
- La librería utilizada está configurada para utilizar la señal de reloj del mando original de la PS2. Debido a que otros fabricantes de estos mandos, que no sea SONY, no utilizan la misma señal de reloj, la librería no es capaz de establecer la conexión.

La solución al primer problema ha sido el uso de un conversor lógico de 3.3V a 5V. Se trata de un componente muy barato y que se puede conseguir por ebay. El segundo problema se ha resuelto de la manera más simple: consiguiendo un mando original SONY de la PS2.

En cuanto a la soldadura de los cables del mando con la *protoshield* también hubo dificultades, ya que en un principio, con los mandos no originales, se abrió el conector del mando para extraer así los cables y soldarlos directamente en la placa. El problema fue primero al abrir el conector, se tuvo que utilizar un soldador para fundir el plástico; y después para soldar los cables ya que al ser muy finos hacen que la soldadura sea muy difícil de hacer. Una vez soldados los cables, las conexiones eran muy frágiles y dejaban de hacer contacto al mínimo esfuerzo que se le aplicaba. La solución fue no abrir el conector y utilizar directamente los pines ya instalados.

2. Modulo RF
   Los primeros módulos utilizados eran una versión antigua utilizada en un proyecto realizado en la UPC, proporcionados por el departamento de Electrónica.
   Siguiendo las instrucciones del blog *arduino-info.wikispaces.com [13]* sobre la conexión del módulo y el uso de la librería, se conectaron mal los pines, ya que no se estaba al corriente de que los esquemáticos mostrados en el blog eran de una versión más actualizada del mismo modulo RF. Este problema causó mucha pérdida de tiempo y frustración, pero finalmente se solucionó comprando la versión más reciente, que como extra disponía de una antena para ampliar el alcance de la transmisión de datos.

6.2.1.2 Mejoras futuras

Las mejoras que deberían realizarse al mando de radiocontrol construido son:

- Substitución de las baterías Ni-MH recargables por baterías Li-PO, ya que al tener una mayor densidad de energía que las primeras permiten ahorrar espacio sin perder autonomía del mando.
- Construcción de una caja contenedora de la placa Arduino y de las baterías, para evitar golpes y también que los cables de alimentación dejen de hacer contacto por el hecho de tener las baterías colgando.
  A esta caja se le podría añadir un gancho para poner una cuerda, para que sea posible que el usuario pueda ponerse la cuerda alrededor del cuello. Esto resultaría muy cómodo para el

usuario, ya que no debería estar preocupándose de donde poner el Arduino y le permitiría concentrarse en el pilotaje del *dron*.
Las opciones de fabricación de esta caja serian por impresión 3D, para hacer la caja fácilmente personalizable, o utilizando planchas de metacrilato, por ser un material muy ligero y barato.

6.2.2 Controlador de vuelo

El controlador de vuelo va montado junto con la estructura física del *dron* y está constituido por:

- Placa Arduino y *protoshield*.
- Modulo RF nRF24L01+.
- Sensor inercial MPU-9250.

Además, es necesario realizar las conexiones a la batería para que se pueda alimentar y a los *ESCs* para enviar las señales *PWM* que controlarán los motores *brushless*.

En este apartado se explicarán los pasos realizados para hacer el controlador de vuelo operacional.

Primero de todo, se ha montado el modulo RF de la misma manera que se explica en el apartado 6.2.1, utilizando la misma *protoshield* y realizando las mismas conexiones.

Después, se ha pasado a la conexión del sensor inercial MPU-9250. Como se ha explicado en el apartado 6.1.3, se utiliza el bus I2C que necesita únicamente 4 cables: SDA, SCL, VCC y GND.

Finalmente, se han conectado también los cables de señal PWM de cada uno de los ESCs con su pin correspondiente en la *protoshield*. Los pines del Arduino elegidos para que generen la señal PWM son el 3, 5, 6 y 7, que se conectan al motor 1, 2, 3 y 4, respectivamente. La elección de estos pines será explicada con más detalle en el aparado Software.

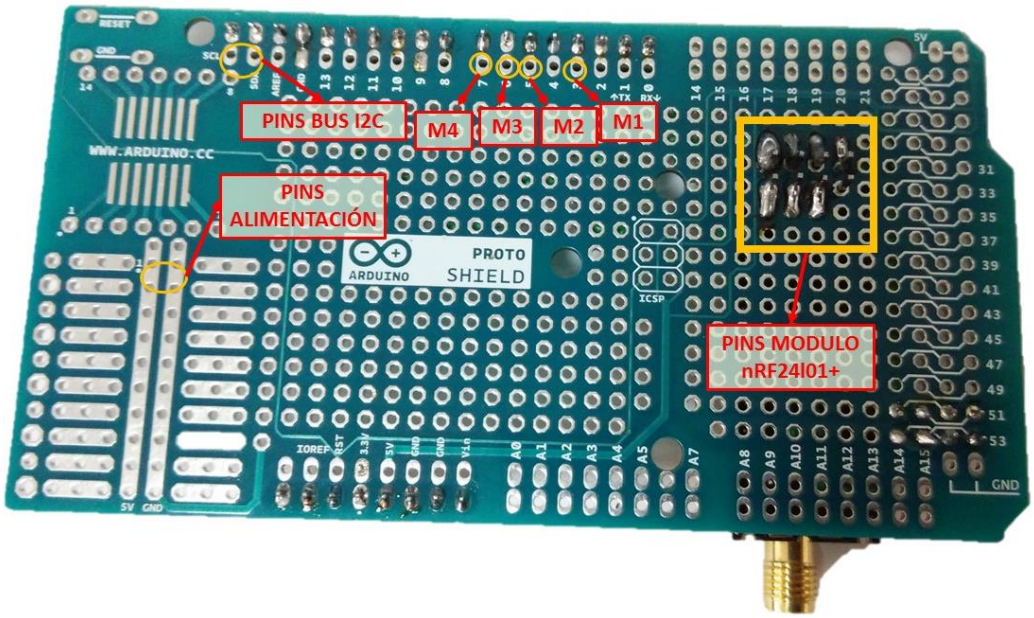

Fig. 80. Conexiones a realizar sobre la protoshield (Fuente: propia)

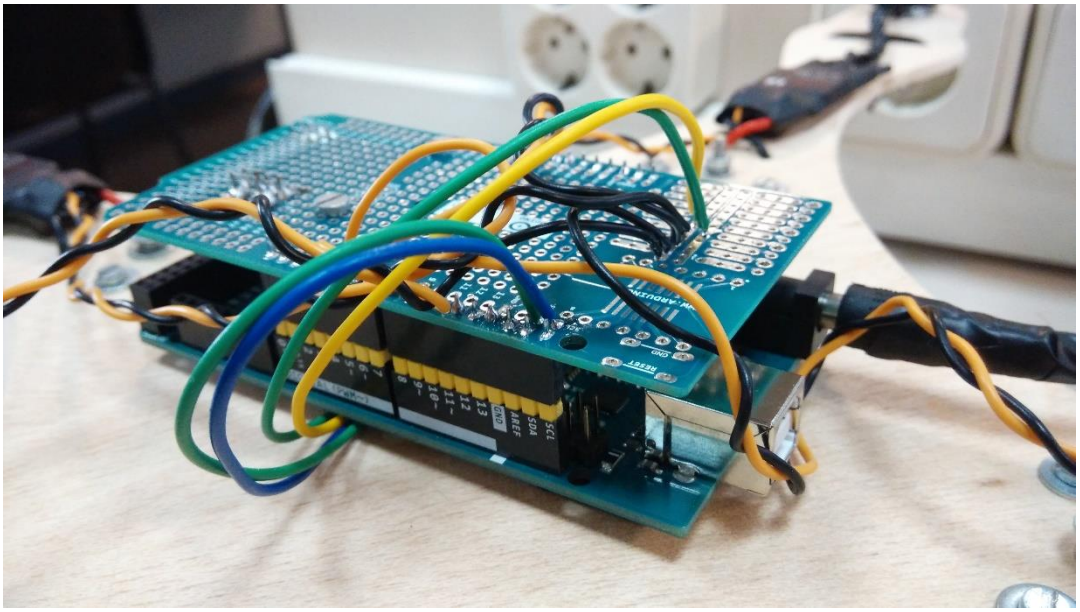

Fig. 81. Arduino y protoshield con los cables soldados (Fuente: propia)

En la Fig. 81 se ve el conjunto montado encima del dron. Los cables que salen por debajo del Arduino son los que se conectan al sensor inercial. Este se ha colocado debajo del Arduino y en el centro geométrico de la estructura por los siguientes motivos:

- Protegerlo de posibles golpes.
- Protegerlo de interferencias electromagnéticas.
- Evitar posibles errores estacionarios de las lecturas por no estar colocado en el centro.

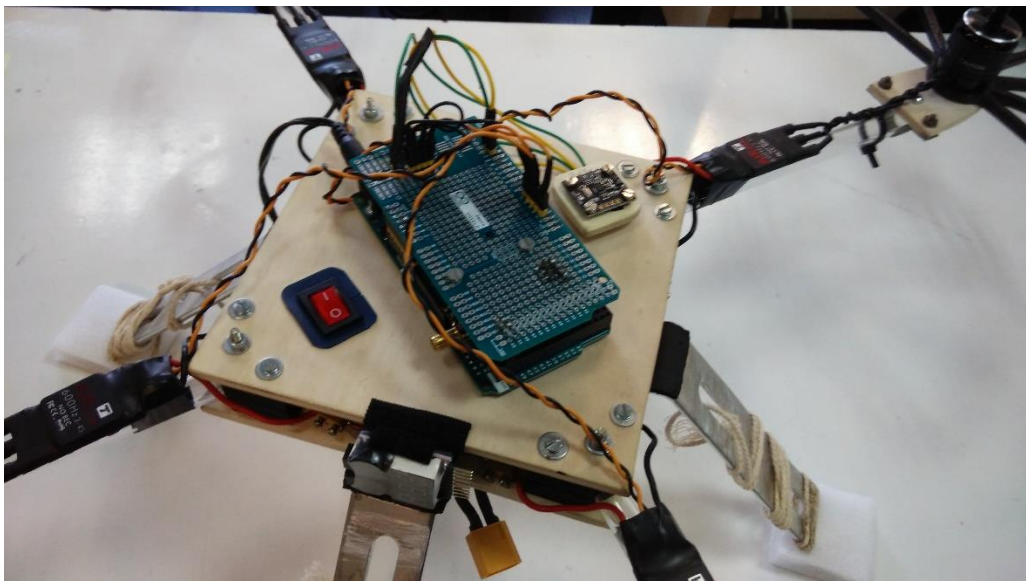

Fig. 82. Montaje con el sensor colocado a la derecha del Arduino (Fuente: propia)

Fig. 83. Montaje con el sensor colocado en el centro (Fuente: propia)

6.2.2.1 Dificultades encontradas

En el montaje del controlador de vuelo se encontró la siguiente dificultad:

- Como primera solución se optó por hacer todos los cables de los componentes extraíbles, lo que equivale a soldar cada cable a un conector hembra y en la *protoshield* soldar tiras de pines para poder colocar los conectores.
  Esta solución se adaptó para poder desmontar el Arduino con su *protoshield* con facilidad.
  El problema surgió después de hacer unas cuantas pruebas, cuando los conectores empezaron a dejar de hacer contacto, lo que resultó en problemas que surgían de manera aleatoria, como por ejemplo los motores dejaban de funcionar sin motivo aparente.
  Finalmente, se decidió soldar directamente los cables a la *protoshield* para eliminar el problema, aunque esta opción dificultó el desmontaje del Arduino de la estructura.

6.2.2.2 Mejoras futuras

Una de las posibles mejoras que se podría aplicar para el controlador de vuelo sería, por ejemplo, el uso de conectores JTS para los cables de los ESCs y los del sensor. De esta manera, se tendrían las ventajas de poder desconectar y así desmontar la placa Arduino con facilidad.

Fig. 84. Conector hembra JST. Fuente: catálogo de productos de Diotronic.

Fig. 85. Conector macho JST. Fuente: catálogo de productos de Diotronic.

7. Software

El software utilizado para el *dron* y para el mando es el resultado de una serie muy larga de pruebas, errores y mejoras realizadas de manera continua durante toda la duración del proyecto.

En un principio se empezó implementando solamente la librería *PSX* para poder leer las señales provenientes de las palancas analógicas y de los botones del mando de la PS2. Seguidamente se incluyó la librería RF24 a los dos Arduino, para que uno enviase las señales leídas y el otro las recibiera correctamente.

7.1 Implementación de las librerías PSX y RF24Network

El primer código utilizado para el uso del mando de la PS2 es un ejemplo incluido en la librería *PSX* que permite aprender las órdenes principales:

- **config_gamepad**(PS2_CLK, PS2_CMD, PS2_SEL, PS2_DAT, true, true);
  Es necesario para establecer los pines del Arduino a los que está conectado el mando y para activar o desactivar la opciones de medir la presión con la cual se aprietan los botones y de vibrar, siendo estas las últimas dos opciones de la orden.
  Devuelve *True* si no ha habido errores.
- **read_gamepad**(false, vibrate);
  Sirve para hacer la lectura del estado del mando y para iniciar o no el motor que produce la vibración del mando.
- **Button**();
  Devuelve el estado del botón especificado en el argumento. Los botones a elegir son:
  - Botones de dirección: PSB_PAD_UP, PSB_PAD_DOWN, PSB_PAD_RIGHT, PSB_PAD_LEFT.
  - Botones de acción: PSB_RED, PSB_GREEN, PSB_BLUE, PSB_PINK, PSB_L3, PSB_R3.
- **Analog**();
  Devuelve un valor entre 0 y 255 correspondiente a la posición de la palanca analógica. Las palancas a elegir son:
  - Derecha: PSS_RY, para el eje Y, y PSS_RX, para el eje X.
  - Izquierda: PSS_LY, para el eje Y, y PSS_LX, para el eje X.

En la Fig. 86 se muestra el esquemático que describe el funcionamiento del programa utilizado y también de la subdivisión típica de un programa de Arduino:

- Antes del *Setup* se declaran todas las variables necesarias para el funcionamiento del programa y también los objetos que se utilizarán.
- En el *Setup* normalmente se inicia la comunicación *Serial* con el PC, aunque no es necesaria para el correcto funcionamiento, y también se inicializan y configuran los objetos creados.
  Esta parte del programa solo se ejecutará una vez, justo después de encender el Arduino.
- En el *Loop* se escriben las acciones que se quieren hacer con las variables y sobre los objetos.
  A diferencia del *Setup* esta parte se ejecutará de manera cíclica hasta que no se apague el Arduino o se produzca algún error.

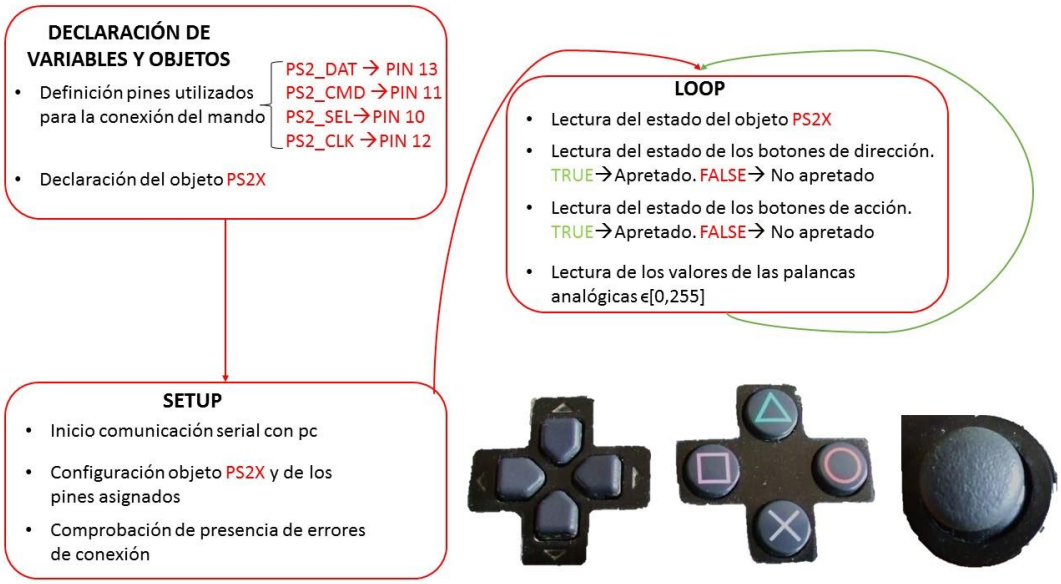

Fig. 86. Esquema del programa para lectura del mando PS2. (Fuente: propia)

Después de solucionar los problemas detectados con la conexión del mando, explicados en el apartado 6.2.1.1, y de comprobar el funcionamiento del mando, se han seleccionado los botones necesarios para el control del dron:

- Palanca analógica izquierda para controlar el *Throttle* y el *Yaw*, siendo el primero asignado al eje Y de la palanca y el segundo al eje X.
- Palanca analógica derecha para controlar el *Pitch* y el *Roll*, asignados de la misma manera que la palanca izquierda.
- Botón L3, que se acciona apretando la palanca analógica izquierda hacia abajo, para armar los motores y para desarmarlos.

El siguiente paso fue implementar la librería RF24Network para transmitir las señales de los botones del mando al dron. De la misma manera que la librería PSX, la librería RF24Network está provista de ejemplos: uno para el envío de información y otro para la recepción. A continuación se describen brevemente las órdenes necesarias:

- **RF24 radio**(pin_CS, pin_CE);
  Constructor que crea el objeto radio y define los pines que se utilizarán para el CS (Chip Select) y el CE (Chip Enable).
- **RF24Network**(radio);
  Es el constructor de la red sobre el objeto radio creado con el constructor anterior.
- **struct payload** {byte ms; byte mess};
  Esta función se utiliza para definir la estructura interna del mensaje que se enviará o se recibirá.
- **SPI.begin**();
  Es necesaria para inicializar el bus SPI.
- **begin**(channel, node_address);
  Inicia la red y define el canal que se utilizara (0 – 127) y la dirección del nodo.
- **update**();

Función a llamar cada vez que se pasa por el *loop* para actualizar la red.

- **available**();
  Función que controla por si hay mensajes nuevos para el nodo. Si hay algún mensaje devuelve *true.*
- **read**(header, &payload, sizeof(payload));
  Lee el mensaje recibido.
- **write**(header, &payload, sizeof(payload));
  Envía un mensaje.

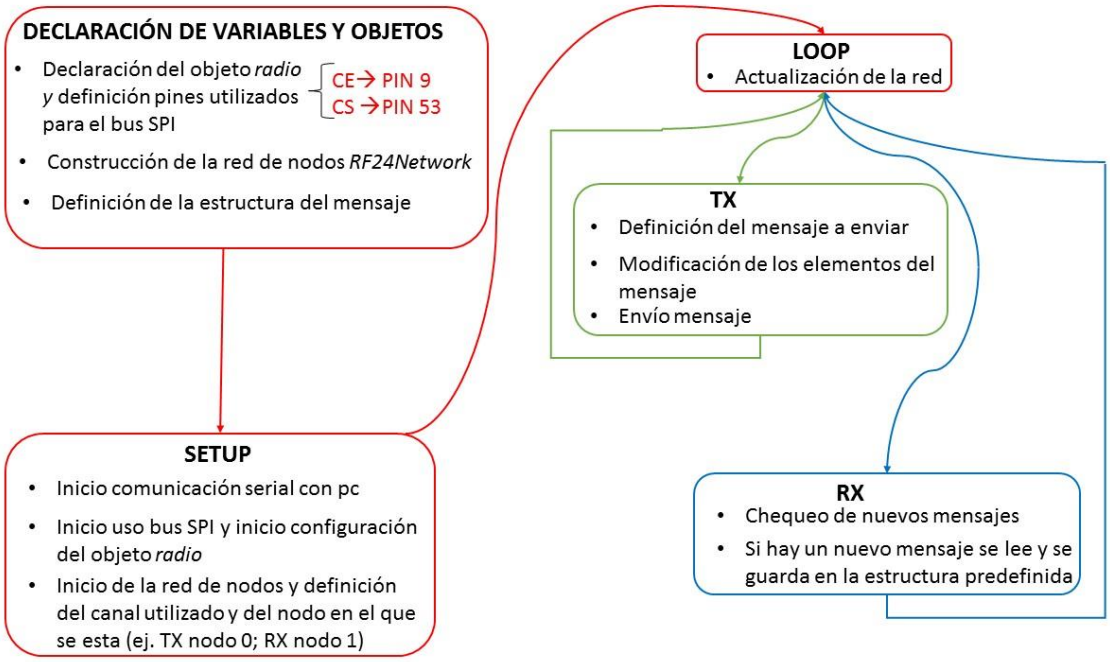

Fig. 87. Esquema del programa para enviar o recibir un mensaje. (Fuente: propia)

7.2 Control de la velocidad de los motores

Como bien se ha explicado en apartados anteriores el control de los motores se hace mediante el uso de señales PWM generadas por el Arduino. En un principio se utilizó la librería que utiliza la placa para generar señales necesarias para controlar motores Servos. La librería en cuestión es la llamada *Servo* y su uso es muy sencillo. Las órdenes necesarias para poder controlar un motor son:

- **Servo** servo1;
  Se utiliza para crear el objeto servo1.
- **attach**(pin);
  Se define el pin en el que está conectado el motor.
- **write**(throttle);
  Se genera la señal PWM proporcional a *throttle*. Para motores continuos, como los que se utilizarán, la variable tiene que estar entre 700 y 2300 para funcionar. Los que se han seleccionado son 700, siendo esto señal mínima, y 2000 señal máxima.

Al utilizar los motores por primera vez, durante la fase de aprendizaje de uso de estos, se hizo uso de un potenciómetro para variar el parámetro *throttle*. Después de varios intentos y viendo que el motor

no reaccionaba se decidió descartar el potenciómetro y controlar la velocidad del motor mediante teclado o directamente mediante el mando de la PS2.

El motivo por el cual el potenciómetro no podía controlar el motor es que el programa diseñado leía directamente la señal analógica generada por el potenciómetro y la enviaba directamente mediante la orden *write*. Esto creaba una incompatibilidad de valores entre la señal analógica y la orden *write*.

Después de decidir utilizar el teclado todo resultó mucho más sencillo. Primeramente se escribió un programa para adecuar los ESCs de tal manera que respondan concorde a las señales PWM enviadas. Esto se hace mediante una secuencia de señales PWM enviadas por el Arduino, especificada por el fabricante, que se debe realizar a cada uno de los ESCs por separado. Antes de la realización de la calibración es necesario quitar las hélices para evitar daños a otras personas y también encender los motores en el momento indicado y no desde el principio, sino el motor no realizará la calibración correctamente.

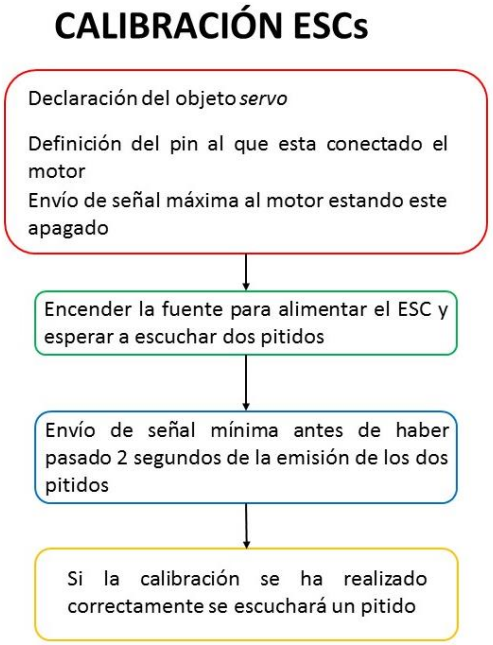

Fig. 88. Calibración de los ESCs. (Fuente: propia)

Una vez realizada la calibración se pueden controlar los motores con un algoritmo que permita controlar la variable *throttle*. Esta variable tendrá como limites los valores utilizados en la calibración. En este algoritmo se ha decido utilizar ya las palancas analógicas del mando. Por lo tanto el programa deberá leer qué valor tiene la variable asociada al eje *Y* de la palanca analógica izquierda (PSS_LY) y adecuarla para ser recibida por el ESC.

La variable *throttle* es una variable de tipo acumulativo, lo que quiere decir que el movimiento de la palanca analógica causa un incremento o decremento de la variable *throttle* y si se deja como está el valor se mantiene. Esto se ha decidido así por cómo funciona la palanca del mando, que tiene unos muelles que hacen que su posición de reposo sea la central o, traducido en el valor que recibe el Arduino, 127 de un rango de entre 0 y 255, y no sea la posición más baja correspondiente a un valor 0.

En la Fig. 89 se muestra el esquema representativo del algoritmo de control de los motores. En la implementación real del programa hay unas condiciones que se deben cumplir para asegurar el correcto funcionamiento:

- La variable throttle_RF que representa la posición de la palanca analógica debe ser de tipo byte para asegurar su correcto envío y recepción.
- La variable throttle, que es la que resulta del procesamiento de throttle_RF, debe pertenecer al intervalo [700,2000] y por ningún motivo salir de este, ya que podría ser peligroso para los usuarios.

- La variable proporcional a throttle_RF también tiene que estar restringida y tiene que tener valores pequeños ya que si se deja crecer demasiado pueden haber cambios bruscos en la velocidad de los motores.

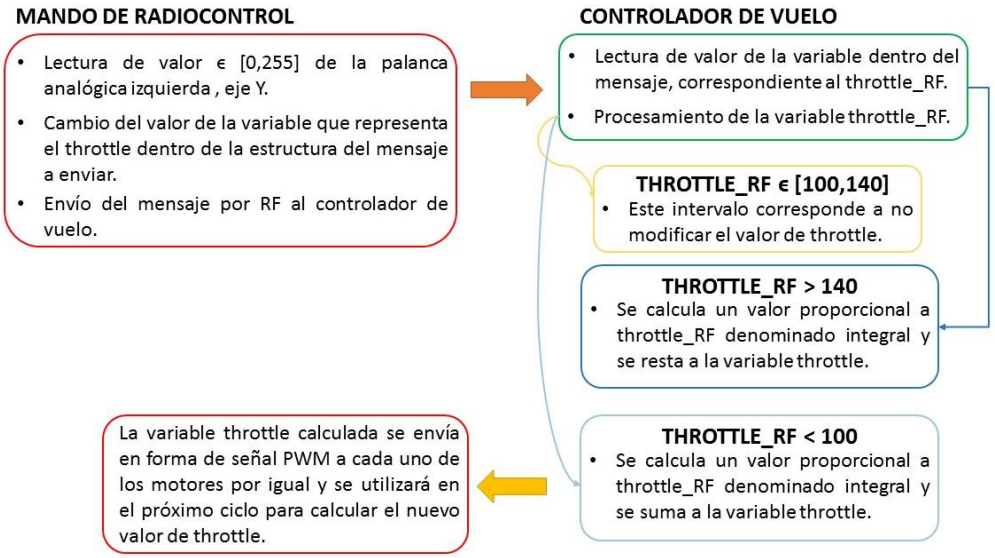

Fig. 89. Esquema del programa para controlar los motores. (Fuente: propia)

7.2.1 Dificultades encontradas

Durante la implementación del algoritmo de control de los motores se ha tenido un problema que ha ido apareciendo de manera aleatoria y a primera vista sin motivo. Este ha sido un funcionamiento incorrecto de la recepción de mensajes por parte del controlador de vuelo, que hacía que el intervalo de tiempo entre un mensaje recibido y otro fuera demasiado grande, causando un crecimiento muy lento de la variable throttle y por consiguiente un cambio muy lento en la velocidad de los motores.

Las posibles causas que se tuvieron en cuenta fueron:

- Agotamiento de las baterías del mando que hacían lento el envío de los mensajes.
- Configuración incorrecta del módulo RF.
- Interferencias electromagnéticas ambientales.
- Incompatibilidad entre la librería RF24 y la Servo.

Después de realizar varias pruebas y búsqueda en internet se aislaron las verdaderas causas:

- Por una parte se descubrió que, en el laboratorio donde se realizaban las pruebas, algunos días había muchas interferencias en los canales utilizados para la transmisión de datos. Esto se comprobó encendiendo solo un módulo RF en modo receptor y viendo que recibía información aleatoria sin tener el otro modulo encendido y enviando.
- Por otra parte, en el fórum de Arduino se encontraron muchas conversaciones referentes a problemas tenidos en el uso de la librería Servo con la librería RF24.

Las soluciones fueron cambiar de canal por el cual se realizaban las transmisiones y escribir unas funciones que substituyeran a la librería Servo, utilizando la función *analogWrite*() que permite generar señales PWM por los pines del Arduino.

7.3 Implementación de la librería RTIMULib

De la misma manera que las otras librería, la RTIMULib contiene ejemplos para entender su funcionamiento básico. Las órdenes necesarias para extraer la información del sensor son:

- **RTIMU** *imu;
  Objeto que representa el sensor.
- **RTFusionRTQF** fusion;
  Objeto en el que se encuentran los valores calculados por el algoritmo de fusión.
- **RTIMUSettings** settings;
  Objeto que contiene las configuraciones del objeto imu y de la librería.
- **Wire.begin**();
  Comando necesario para empezar la comunicación con el sensor.
- imu = **RTIMU::createIMU**(&settings);
  Se genera el objeto imu.
- **fusion.setSlerpPower**(0.02);
  Controla el algoritmo de fusión. Si el valor indicado es 0 se utilizan solo los giroscopio para hacer el cálculo de los ángulos, y si es 1 solo se utilizan los acelerómetros y magnetómetros. Un valor intermedio combina los 3 sensores.
- **fusion.setGyroEnable**(true);
  fusion.setAccelEnable(true);
  fusion.setCompassEnable(true);
  Habilita el uso de los sensores en el algoritmo de cálculo.
- imu->**IMURead**();
  Lee los últimos datos disponibles.
- **fusion.newIMUData**(imu->getGyro(), imu->getAccel(), imu->getCompass(), imu->getTimestamp());
  Calcula la orientación del sensor.
- **RTMath::displayRollPitchYaw**("Pose:", (RTVector3&)fusion.getFusionPose());
  Imprime por el serial la orientación calculada.

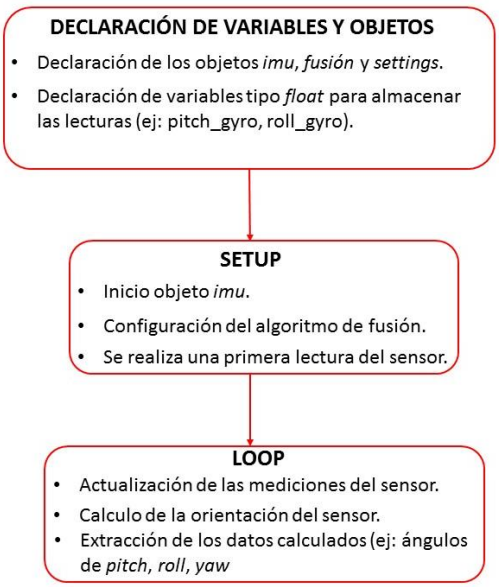

Fig. 90. Esquema del programa para obtener la orientación del sensor. (Fuente: propia)

7.4 Implementación de la librería PID

El software del dron necesita de una herramienta para poder actuar sobres los motores, concorde a las órdenes del piloto a través del mando y en función de la orientación que tiene el dron en el instante. Esta herramienta tendrá que comparar las órdenes del mando con las mediciones realizadas por el algoritmo de fusión, y en función de esta comparación generar señales PWM que resulten en un estado del dron tal como las ordenes manda.

Lo que se está describiendo hasta ahora es la función que hace un PID, que es un algoritmo de control de lazo cerrado que calcula el error entre un input medido (orientación) y una consigna deseada (órdenes del mando) e intenta minimizarlo actuando sobre el sistema por medio de una variable output.

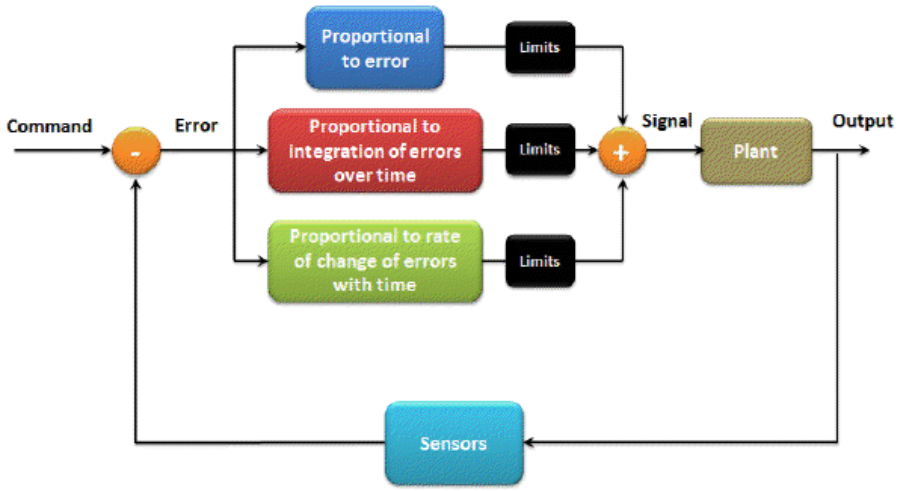

Fig. 91. Diagrama de la estructura de un PID. Fuente: www.autoquad.org.

Para implementar un PID en el software se ha utilizado la librería propia de Arduino que permite utilizar tales algoritmos de una manera muy intuitiva.

En el software diseñado se necesitará tener 2 PIDs por cada eje de rotación del *dron*: el primer PID calculará el error entre el ángulo, en grados, en el que se encuentra cierto eje y la orden del mando que corresponde al ángulo deseado en el mismo eje, también en grados. A partir de esta diferencia generará una variable de salida (output) que indicará a qué velocidad tiene que rotar el eje. El segundo PID calculara el error entre la variable calculada por el primero, que tendrá unidades de grados partido segundos, y la medida de la velocidad angular del eje en ese instante y generará una variable output que indicará lo que hay que sumar o restar a la señal que actúa sobre la pareja de motores correspondiente al eje.

Fig. 92. Estructura del PID por cada eje. (Fuente: propia)

7.5 Estructura del algoritmo de control de vuelo

Una vez explicado brevemente el uso de cada uno de los componentes se pasa a ilustrar como es el algoritmo que funcionará en el dron durante el vuelo.

La estructura del algoritmo tiene diferentes partes almacenadas una en cada archivo para que pueda ser más sencillo entender su funcionamiento y realizar modificaciones:

- RX_NETWORK_IMU_PID.ino
  Este archivo contiene el programa principal del dron en el cual está el *loop* que se ejecutara de manera cíclica durante el vuelo.
- IMU.ino
  Contiene las líneas de código necesarias para habilitar el funcionamiento del sensor. Solo se ejecuta una vez durante el *setup* del programa principal.
- PID.ino
  Es necesario para inicializar los PID que el programa principal utilizará para hacer los cálculos. También contiene una función para poder cambiar los valores de Kp, Ki y Kd de los PID durante el vuelo.
- MOT.ino
  Contiene las funciones necesarias para: armar los motores, esta función se ejecutara solo una vez durante el *setup* del programa principal, y para enviar una señal PWM a los motores, que se ejecutara cada vez que se quiera cambiar la velocidad de los motores.
- Config.h
  Este archivo contiene constantes cuyo valor no cambia durante el vuelo y que sirven para el funcionamiento del programa principal. En este están almacenados por ejemplo los pines a los que van conectados cada motor, los límites de algunas variables, etc.
- adec_Throttle.ino; adec_Roll.ino; adec_Pitch.ino; adec_Yaw.ino.
  Cada uno de estos archivos contiene una función para adecuar su respectiva variable recibida del mando para que pueda ser utilizada por los PID para hacer cálculos.

A continuación se muestra el esquema que ilustra el funcionamiento del programa principal y todas las fases por las que pasa.

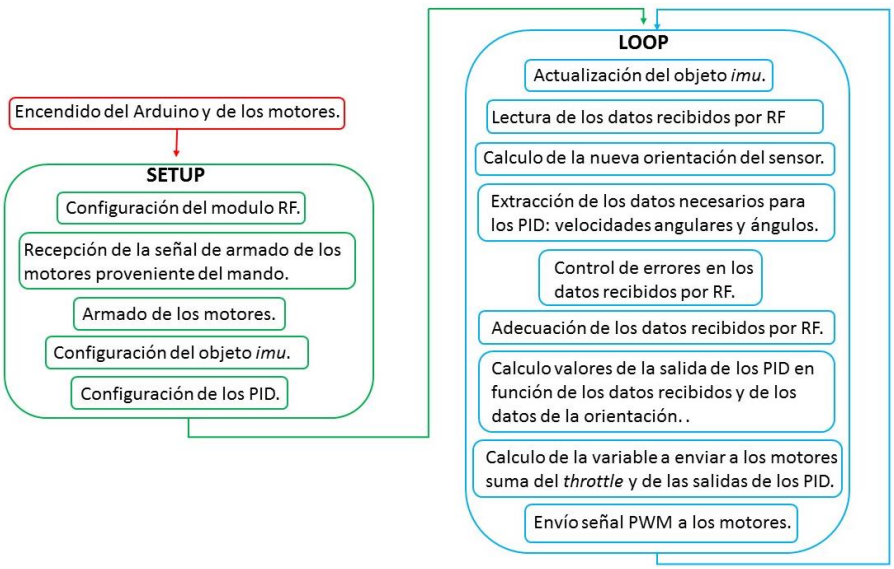

Fig. 93. Esquema de funcionamiento del programa principal. (Fuente: propia)

7.5.1 Pruebas de calibración del algoritmo de vuelo

Antes de poder realizar un posible vuelo con el dron es necesario hacer la calibración de algunos valores internos al algoritmo de cuyo valor dependerá la estabilidad del cuadricóptero. La mayoría de estos valores se encuentran reunidos todos en el archivo *Config.h,* para hacer agilizar su modificación. Entre estos valores se encuentran:

- Los valores máximos de los ángulos de Roll y de Pitch, denominados como *int_pitch_MAX* e *int_roll_MAX* que definirán la inclinación máxima que se demandará al dron durante el vuelo. En un principio se quieren mantener estos dos valores iguales y bajos, entre 15.0 y 20.0 grados, para que los cambios en la orientación no sean muy grandes. Otra cosa a tener en cuenta es que estos valores deberán coincidir en el algoritmo del mando y en el de vuelo para evitar incidencias.
- El valor máximo de cambio del *throttle*, denominado *int_MAX*. Este valor define el salto máximo entre un valor y otro de la velocidad giro de los motores, velocidad de cambio de la variable *throttle*.
  En el archivo *Config.h* también se encuentra un valor denominado int_MAX_2, esto es porque se ha decidido dividir en dos tramos la variable throttle: en el primer tramo su velocidad de cambio es más alta ya que en este momento el dron se encuentra despegando o aterrizando, por lo tanto se necesitan correcciones rápidas en la velocidad de giro de los motores. En el segundo tramo el dron ya no se encuentra en una zona de riesgo y no es necesario tener cambios bruscos
- Los valores que definen los PID: la Kp, Ki y Kd y los límites de actuación que tiene cada uno de los PID.
  El algoritmo de vuelo utiliza 6 PID, 2 por cada eje de rotación. Como se ha descrito anteriormente cada eje tiene un PID que controla el ángulo de inclinación y otro que controla la velocidad de giro colocados en serie. Que estén colocados en serie significa que la variable de la salida del primero será la consigna del segundo. Este hecho es muy importante ya que significa que si el controlador de velocidad no está calibrado correctamente, el control del ángulo en cualquier caso no funcionará de una manera satisfactoria.
  Los límites de actuación de los PID se consiguen calibrar de una manera relativamente sencilla: durante las primeras pruebas de calibración se deja un margen pequeño hasta conseguir una respuesta del sistema satisfactoria, y después ya se puede aumentar ligeramente el poder de actuación del PID.
  Para la calibración de los valores de Kp, Ki y Kd ya no es tan sencillo y el procedimiento a seguir es explicado en el siguiente apartado.

7.5.1.1 Calibración de los PID

La calibración de los valores característicos de los PID es una tarea muy complicada ya que estos dependen de muchas variables físicas del modelo y no hay una manera teórica de fácil desarrollo con la cual se puedan calcular.

Para la determinación de estos valores hay dos posibles vías a seguir:

- La definición de un modelo matemático que permita simular el funcionamiento del dron, teniendo en cuenta todas las variables físicas y haciendo el menor número posible de

simplificaciones. Una vez obtenido el modelo crear un algoritmo de control, no necesariamente basado en PID, que mejor se adapta al modelo. Conseguido el algoritmo de control se tienen que realizar simulaciones con el Matlab para conseguir valores aproximados característicos del algoritmo de control, que serán la base de partida para la realización de experimentos con la planta física, el dron, para conseguir los valores que mejor se adapten.
El apartado 3 es el principio, de este estudio al cual todavía falta conseguir el algoritmo de control que mejor se adapte. Esta opción es quizá la que mejor resultados consiga una vez implementada, aunque supone una carga de trabajo muy grande y tiempo del cual no se dispone, ya que se recurrió a esta opción a poco tiempo de la entrega de este trabajo.

- La experimentación directa con el algoritmo de control realizado sin diseñar un modelo matemático, sino realizar la calibración desde cero con la planta física. Esta experimentación consiste en dejar libre un eje de rotación del dron y anclar el otro. De esta manera se consigue que solo haya un movimiento de rotación, permitiendo calibrar el PID del eje correspondiente. Este experimento solo es válido para los ejes de Roll y Pitch, que son los que suponen problemas a la hora de calibrar.

La vía elegida ha sido la segunda dejando la primera empezado y para acabar posiblemente en el trabajo de final de máster.

Para la realización de la experimentación ha sido necesario la creación de un banco de pruebas seguro, que pusiera al mínimo las probabilidades de hacer daño a personas, y que mejores resultados permitiera obtener. Así se decidió utilizar mesas presentes en el laboratorio utilizado tal como muestra la Fig. 94.

Fig. 94. Banco de pruebas. (Fuente: propia)

Con esta primera iteración del banco de pruebas se tuvieron ciertos problemas al principio en tema de seguridad y libertad de movimiento del dron. Por este motivo, se cambió la orientación de las mesas tal y como muestran la Fig. 95 y la Fig. 96, permitiendo así atar el dron a las mesas con cuerdas para impedir que saltara volando del banco y también mayor libertad de movimiento de rotación respeto a un eje y menor fricción con los puntos de apoyo.

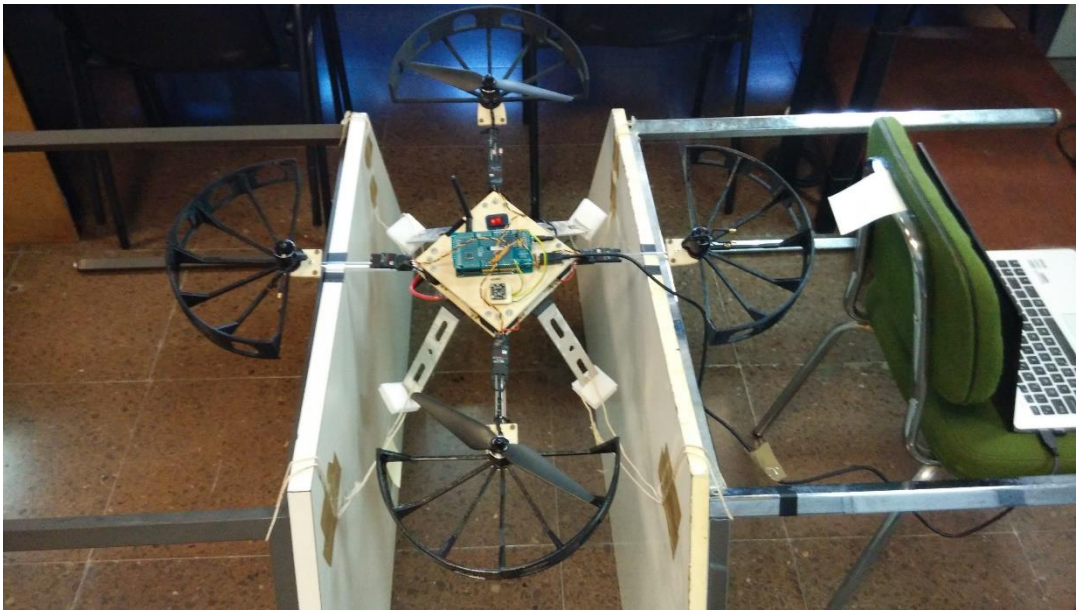

Fig. 95. Banco de pruebas mejorado. (Fuente: propia)

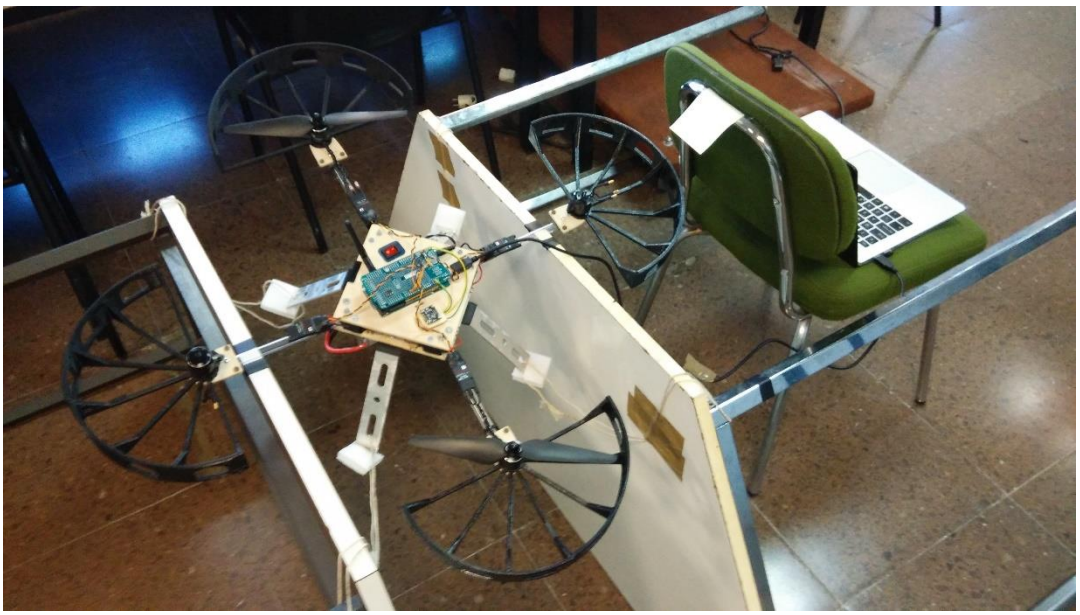

Fig. 96. Banco de pruebas mejorado. (Fuente: propia)

Después de tener un banco de pruebas valido es necesario escribir líneas de código que permitan modificar los valores característicos de los PID e imprimirlos por el Serial del Arduino, conectado al PC, en tiempo real. La función realizada es la *readAndChange(),* que se encuentra en el file *PID.ino* y permite hacer exactamente lo descrito anteriormente utilizando las teclas numéricas del PC para aumentar o disminuir los valores.

Las precauciones a tomar antes de realizar el experimento son:

- Atar unas cuerdas a las patas del dron y pasarlas por las patas de la mesa y asegurarlas con un poco de cinta, de manera que no dejen demasiada libertad de rotación y que estén bien tensas cuando el dron está inclinado al máximo deseado.

- Quitar las hélices de los motores del eje que no se quiere calibrar y desconectar los cables de alimentación de dichos motores.
- Colocar el cable USB conectado al Arduino de tal manera que no toque las hélices al rotar el dron.
- Siempre estar presentes dos personas durante el experimento: una que controla los datos obtenidos en el ordenador y otra que utiliza el mando de radiocontrol. Ambas con protección en las manos y en la cara.

Una vez tomadas estas precauciones se pasa a la realización del experimento en sí que se compone de los siguientes pasos:

- Encendido y armado de los motores
- Aumento gradual de la velocidad de los motores, previamente limitada mediante software para impedir que el dron pueda volar, hasta llegar al máximo permitido cerca de los valores de rotación en vuelo, ya que el comportamiento cambia a bajas velocidades.
- Aumentar gradualmente el valor de la Kp hasta primero ver se obtiene una respuesta a las órdenes del mando y después hasta obtener oscilaciones.
- Dejar el valor de la Kp en uno que supongo una situación de oscilaciones muy pequeñas o en el mejor caso nulas. Que el eje no se quede horizontal no supone un problema ya que este error se corregirá con la introducción de Ki.
- Aumentar gradualmente el valor de Ki hasta que el error estacionario mencionado anteriormente desaparezca pero no se introduzcan nuevas oscilaciones.
- Si se nota que la respuesta tiene un cierto retraso es posible introducir también Kd, aunque se tiene que tener cuidado ya que aumenta de manera significativa el ruido presente en el sistema.
- Finalmente se pueden ajustar los límites de actuación del PID para obtener una mejor respuesta.[12]

Esta guía se ha extraído de la guía sobre ajuste de PID de la wiki de la página web de autoquad.org.

Los pasos anteriormente mencionados son para calibrar el controlador de ángulo, para calibrar el de velocidad los pasos son los mismos pero hay que observar el comportamiento de la velocidad.

En la Fig. 97 se ilustran los resultados obtenidos después de la calibración del PID de velocidad y solo teniendo este activado, lo que significa que el piloto solo es capaz de controlar con el mando la velocidad de rotación del eje y no el ángulo al que se encuentra. En este se ve la comparación entre las órdenes del mando, en azul, y las mediciones realizadas por el giroscopio de la velocidad real. Como se aprecia hay oscilaciones en la respuesta pero estas no suponen un gran problema ya que son oscilaciones de velocidad y se traducen en oscilaciones muy pequeñas en la posición una vez se manda velocidad cero por parte del mando.

Los valores característicos del PID utilizados para obtener los resultados de los gráficos de la Fig. 97 son los siguientes:

$$Kp = 0{,}95 \quad Ki = 1{,}14 \quad Kd = 0{,}007$$

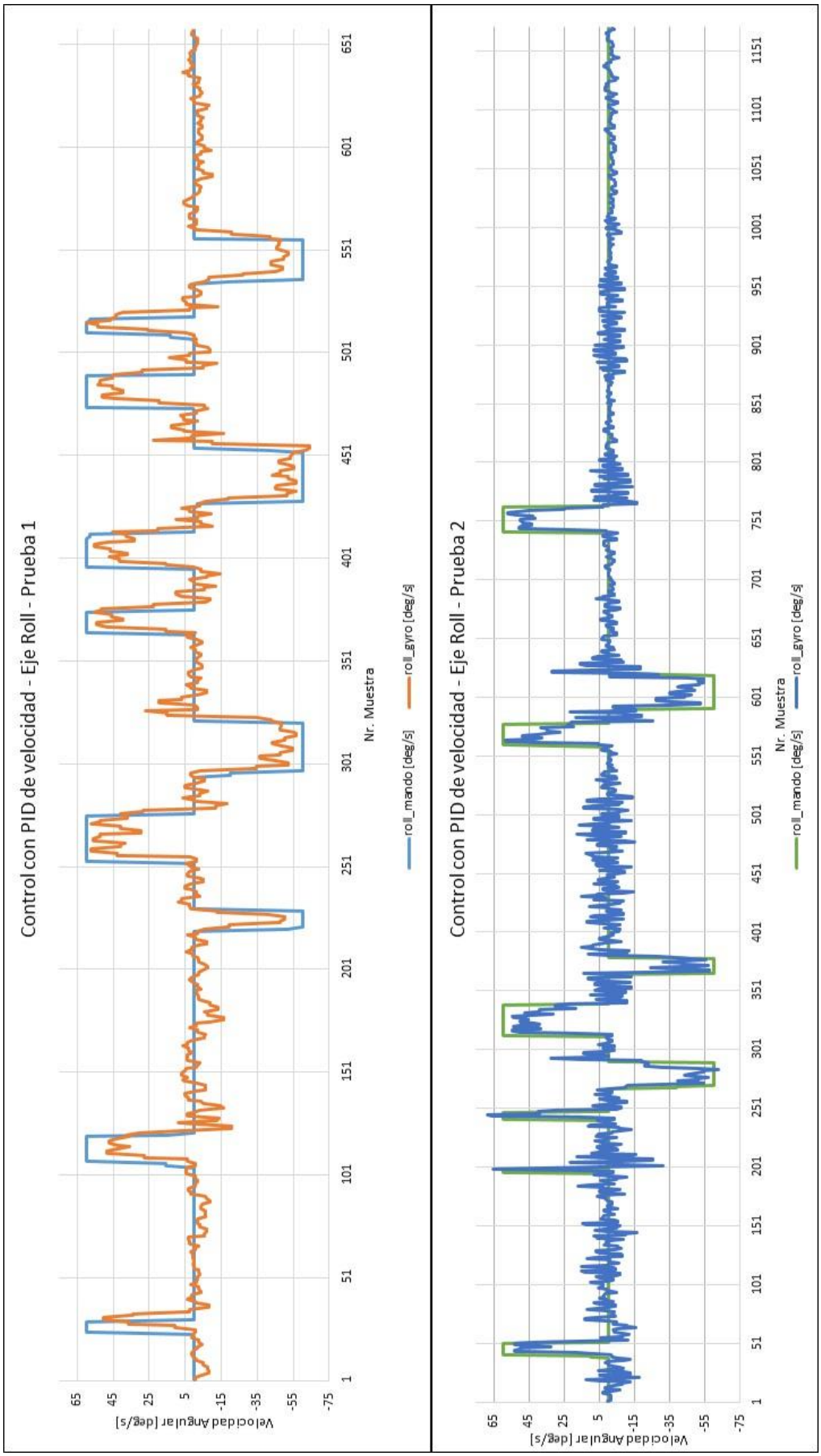

Fig. 97. Gráficos comparativos entre la consigna velocidad y la respuesta real. (Fuente: propia)

Este método de control, por velocidad de giro, es ampliamente utilizado por pilotos experimentados, pero muy difícil de utilizar por pilotos novatos y que tienen poca o nula experiencia de vuelo y acaba siendo la causa de incidentes graves. Aun así se decidió realizar el primer vuelo de prueba con esta configuración, decisión muy poco sabia y tomada vistos los buenos resultados obtenidos en el banco de pruebas. En la Fig. 98 y Fig. 99 se pueden ver las instantáneas del video realizado durante el primer vuelo y en la Fig. 100 el final del primer vuelo.

Fig. 98. Instantáneas del primer vuelo. (Fuente: propia)

Fig. 99. Instantáneas del primer vuelo. (Fuente: propia)

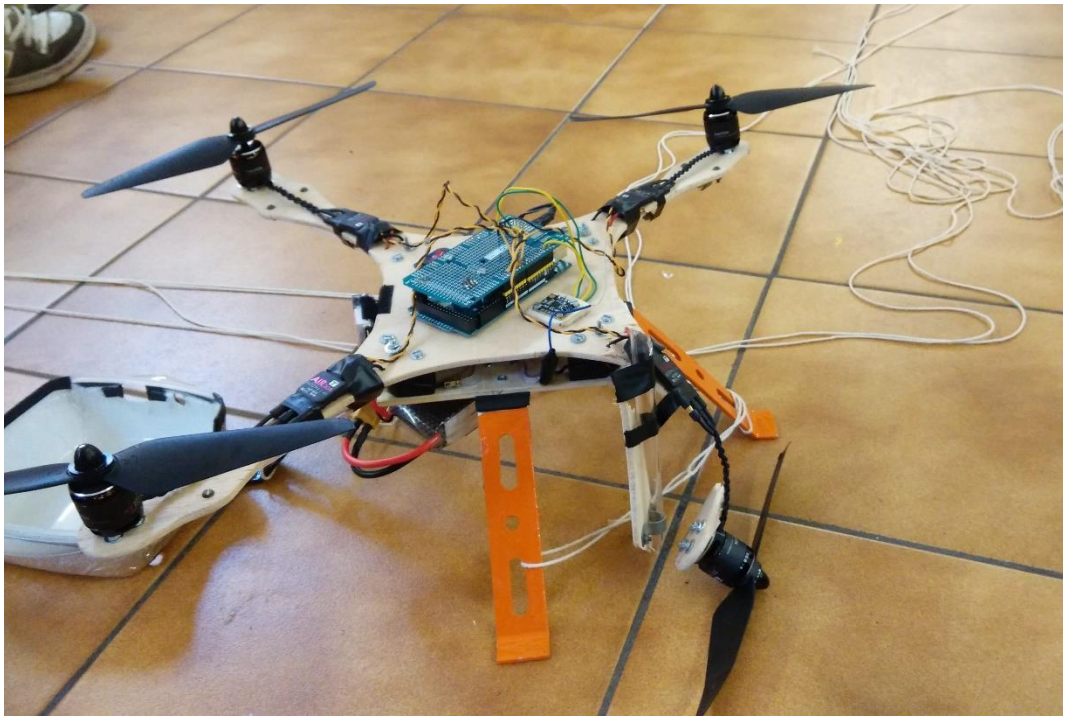

Fig. 100. Final del primer vuelo. (Fuente: propia)

La Fig. 100 muestra claramente los resultados que se obtienen cuando se toman decisiones sin meditar primero las posibles consecuencias.

Aunque se ha confirmado lo dicho anteriormente, que el control de velocidad no es apto para pilotos novatos y que se requiere de implementar el control de vuelo de posición para así hacer la tarea del piloto mucho más fácil.

La calibración del PID de posición se ha realizado varias veces, pero no ha llegado a mostrar resultados igual de precisos como el control de velocidad, lo que requiere ciertos cambios en el controlador de vuelo y quizá también en la estructura. Los cambios que se podrían hacer, en un trabajo futuro, son los siguientes:

- En el controlador aumentar la velocidad de ejecución del programa, que ahora se ejecuta a 90-100Hz. La manera de aumentarla seria cambiar de microcontrolador ya que el poco poder de procesamiento no permite ejecutar el programa más rápidamente.
- En la estructura incluir sistemas de amortiguación para disminuir las vibraciones creadas por los motores, que crean ruidos en las medidas tomadas por la placa IMU.

8. Impacto medioambiental

8.1. Directiva RoHS

Los componentes eléctricos y electrónicos utilizados para la construcción del dron se ven sometidos a la directiva RoHS que se presenta a continuación:

RoHS (Restriction of Hazardous Substances) - Directiva 2002/95/CE, que limita el contenido de sustancias nocivas:

1. *plomo (Pb)*
2. *mercurio (Hg)*
3. *cadmio (Cd)*
4. *cromo hexavalente (cromo VI o Cr6 +)*
5. *bifenoles polibromados (PBB)*
6. *éter difenol polibromado (PBDE)*

PBB y PBDE se utilizan como aditivos - retardadores de fuego en la producción de plásticos

Directiva RoHS (Restriction of use of Hazardous substances – Restricción en el uso de Sustancias Peligrosas) regula el uso de plomo y otros componentes potencialmente peligrosos en aparatos eléctricos y electrónicos. El 1 de Julio de 2006 la Directiva 2002/95/CE entró en vigor en toda la Comunidad Europea. El objetivo de la Directiva - limitar la aplicación de estas seis principales sustancias peligrosas en aparatos eléctricos y electrónicos y proporcionar un nivel de protección para la salud humana y el medio ambiente.

[…]

La directiva RoHS establece límites precisos de los niveles aceptables, y cuya observancia es obligatoria.

La Directiva RoHS se aplica a las siguientes categorías de productos: electrodomésticos, equipos de telecomunicaciones y equipos de tecnología de la información, electrónica de consumo, aparatos de alumbrado, herramientas eléctricas, juguetes, productos de ocio y deportivos, máquinas expendedoras, l bombillas, etc.[15]

Como bien explica la directiva el objetivo principal es proteger la salud humana y el medio ambiente de sustancias peligrosas y nocivas presentes en aparatos eléctricos y electrónicos.

Los componentes principales tales como el Arduino Mega 2560, el sensor MPU-9250, los motores Air Gear 350, el módulo nRF24L01+, las baterías LiPo y el mando de la Playstation 2 siguen con la normativa RoHS ya que son distribuidos y vendidos en la Unión Europea, que obliga a que esta normativa se cumpla. Además los embalajes son provistos de etiquetas RoHS que reflejan el seguimiento de la directiva.

8.2. Reciclaje y eliminación de materiales y componentes

El *dron* construido está constituido por diferentes componentes hechos a partir de diversos materiales. A continuación se listan los principales materiales de los que el *dron* dispone:

- Aluminio
  El aluminio es un metal que permite ser reciclado de manera indefinida, ya que no daña la estructura del material. Además la producción de aluminio utilizando metal para reciclaje resulta ser mucho más barata que la producción del mismo a partir de bauxita, mineral del cual se extrae alúmina y después por electrolisis se produce aluminio.
- ABS y Metacrilato
  Ambos plásticos pueden reciclar de tres maneras: reciclaje mecánico, reciclaje para recuperación de energía (combustión del material), reciclaje termoquímico.
  Al no tener ningún tipo de sustancia peligrosa, los materiales que conforman las protecciones y los brazos del *dron* podrán ser reciclados por reciclaje mecánico, que es el que menos contaminación produce.
- Madera
  El reciclaje de la madera es un proceso necesario para reducir la tala de árboles, que es una práctica que destruye ecosistemas. Es un proceso limpio y económico que no necesita de tratamiento previo: la operación principal de este proceso es la trituración de la madera para producir serrín, que será utilizado para fabricar tableros aglomerados o será utilizado para producir energía eléctrica.

En cuanto al reciclaje de todos los componentes electrónicos, el proceso por el cual se pasa es más complicado que para los materiales anteriormente mencionados. El tratamiento que suelen recibir los residuos de aparatos eléctricos y electrónicos (RAEE) tiene los siguientes pasos:

- Reutilización o Reparación. La opción más simple es de poder reutilizar partes de los aparatos para la fabricación de nuevos, para así disminuir el impacto ambiental producido por la fabricación de partes completamente nuevas. Un ejemplo es el uso que se hace del hierro, que es reutilizado en la industria automovilística.
- Reciclado. Durante el reciclaje se desmontan y se descontaminan los aparatos. Finalmente se recuperan materiales como plásticos, metales o vidrio para ser reciclados.
- Valorización energética. Materiales que no pueden ser reciclados, como ciertos tipos de plásticos, se usan para producir energía para otros procesos.
- Eliminación. Si los aparatos o componentes no pueden utilizarse para ninguna de las opciones anteriores se pasa a hacerles un tratamiento para la reducción de residuos.

Para que un componente pueda recibir el tratamiento explicado deberá ser llevado a al Punto Limpio más cercano y el ayuntamiento se encargará de llevar a cabo su correcto reciclaje.

9. Presupuesto económico

Los autores de este proyecto no han llegado a la solución final de manera directa, como casi nunca sucede en los proyectos de desarrollo tecnológico. En el transcurso del proyecto han sucedido problemas al romperse diversos componentes del cuadricóptero de manera que en este capítulo se consideraran todos los costes y capital invertido que ha supuesto llevar a cabo el proyecto además de los costes por tiempo invertido de los autores del proyecto.

9.1. Costes de material fungible

En la siguiente tabla se muestra el coste de los diferentes componentes que conforman el *dron* y los que han sido usados (y algunos desechados) durante el desarrollo de este proyecto:

Componente	Precio unitario €	Cantidad	Precio €
Motor EMAX MT2213-935KV + pareja hélices 10*4,5	24,0000	1	24,00
Piezas aluminio	22,8900	25%	5,72
Arduino compatible Mega 2560 + cable USB	20,0000	4	80,00
Sensor IMU 9DOF - MPU9250	22,9000	2	45,80
ESC RCI 20A	15,8000	1	15,80
T-Motor AIR GEAR 350 Set	110,7000	1	110,70
Level converter	2,6000	1	2,60
ABS 3mm 1kg Black	22,9300	50%	11,47
Sistema de comunicación NRF24L01	19,5000	2	39,00
Brazos PMMA	0,4725	16	7,56
T-Motor AIR GEAR 350 Set (sin ESC)	72,9000	1	72,90
Portapilas estación RF	3,0000	1	3,00
8 Pilas recargables	1,2488	6	7,49
Baterías LiPo Floureon	31,8750	2	63,75
Alarma batería baja	8,3900	1	8,39
Tablon madera	5,4000	1	5,40
Mando PS2	7,9000	1	7,90
Subtotal			511,48

Tabla 10. Tabla de los costes de los materiales fungibles de este proyecto. (Fuente: propia)

9.2. Costes de material no fungible

En el desarrollo de este proyecto han sido usados materiales y equipos de los cuales se ha calculado un coste de amortización y uso porcentual de materiales. Esto incluye el gasto en componentes electrónicos como cables, regletas, sensor óptico, potenciómetro y protoboard. En cuanto a materiales de ferretería se incluyen herramientas, tornillos, tuercas, arandelas, gomas anti vibraciones, pinturas, etc. También se invirtió en herramientas y materiales de ensamblado como el soldador y la pistola de silicona caliente. De todo lo invertido no se ha utilizado su totalidad y de ahí el desglose de gastos. Los equipos más usados durante el desarrollo de este proyecto son los PC's de los autores del proyecto, fuentes de alimentación, osciloscopio y multímetro.

Concepto	Precio unitario €	Amortización	Precio €	Descripción
Taquilla	33,0000	100,0%	33,00	
Soldador, estaño, lupa	28,6500	70,0%	20,06	
Componentes electrónica	42,3000	80,0%	33,84	Cables, connectores, regletas…
Componentes ferretería	14,5200	90,0%	13,07	Tornillos, tuercas, arandelas…
Cargador baterías LiPo	48,4500	20,0%	9,69	
Cargador baterías AA	5,0000	10,0%	0,50	
Banco de pruebas	18,0000	100,0%	18,00	
Pintura acrílica	9,5000	5,0%	0,48	
Fuente de alimentación	900,0000	2,5%	22,50	
Osciloscopio	525,0000	2,5%	13,13	
Multímetro	75,7750	2,5%	1,89	
PC's	500,0000	10,0%	50,00	
Subtotal			216,15	

Tabla 11. Tabla de los costes de los materiales no fungibles de este proyecto. (Fuente: propia)

9.3. Costes de desarrollo del proyecto

Se calcula que cada día de trabajo en este proyecto la media de horas invertidas ha sido de 6 horas diarias a razón de 30 euros la hora, precio estándar de un ingeniero superior.

El proyecto se puede dividir en diferentes fases: de investigación, diseño, fabricación y pruebas de funcionamiento. A continuación se expone un resumen en la tabla 11.

Fase del proyecto	Horas dedicadas	Precio unitario €	Precio total €
Estudio previo e investigación	114	30	6840
Diseño del cuadricóptero	204	30	12240
Fabricación del cuadricóptero	84	30	5040
Pruebas	180	30	10800
Subtotal	582		34920

Tabla 12. Tabla de los costes de tiempo invertido en el proyecto. (Fuente: propia)

Días de trabajo al mes	
Noviembre	4
Diciembre	8
Enero	7
Febrero	17
Marzo	17
Abril	13
Mayo	16
Junio	15
Total	97

Tabla 13. Resumen de días invertidos en este proyecto. (Fuente: propia)

9.4. Resumen de costes

En la siguiente tabla se presenta el resumen de costes según concepto y, finalmente, añadiendo el 21% de IVA estipulado por ley.

Concepto	Coste €
Material fungible	511,48
Material no fungible	216,15
Tiempo de desarrollo	34920,00
Subtotal	35647,63
21% IVA	7486,00
Total	43133,63

Tabla 14. Costes totales de este proyecto. (Fuente: propia)

10. Conclusiones

La valoración que merece este proyecto, por los autores del mismo, es un poco discordante: por una parte el resultado obtenido es muy satisfactorio y bueno, ya que al final se ha obtenido una plataforma en la cual se puede trabajar en un futuro para conseguir desarrollar el software de vuelo robusto deseado. Por otra parte, no gusta quedarse con la miel en los labios en cuanto se refiere a hacer volar el cuadricóptero.

Aun así, los objetivos principales de este proyecto se han cumplido y con creces. Además de la plataforma compuesta por *dron* y estación de radiofrecuencia, se ha empezado a investigar y desarrollar el software de control de vuelo. Se ha trabajado con sensores, con señales de comunicación y con el procesado de datos, algo con lo que no se contemplaba trabajar durante los meses que ha durado el proyecto.

Los resultados obtenidos no siempre han sido los correctos y en muchas ocasiones ni se han obtenido. Es muy frustrante dedicar horas de trabajo y dinero para que en muchos momentos durante el desarrollo del proyecto no se encontrara solución a los problemas. Pero siendo optimistas, se ha aprendido a que no todo sale a la primera, ni a la segunda e incluso puede que ni a la tercera.

En cuanto al modelado matemático del *dron,* cabe destacar el trabajo de análisis desempeñado. Sin embargo, falta realizar validaciones e intentar no hacer tantas simplificaciones que hagan que el modelo se aleje de la realidad. Este trabajo también podría ser objeto de un trabajo de investigación y desarrollo amplio y extenso.

La parte de la estructura mecánica puede que sea la que deja más orgullosos a los autores de este proyecto: tras pasar por diferentes etapas y siempre teniendo en cuenta los recursos de los que se disponía (y se dispone), la estructura final y el diseño elegido es muy satisfactorio. Así pues, el trabajo desarrollado como mecánico proyectista ha sido muy positivo.

En cuanto a la selección de motores y componentes electrónicos del *dron,* lo único a objetar es el elevado coste invertido en ellos. Sin ningún patrocinio ni inversión más que la individual, puede que no hiciera falta elegir lo mejor del mercado. El resultado, sin embargo, siempre ha sido bueno en cuanto a la respuesta de cada componente: las baterías, los motores y el sensor no merecen ninguna queja.

Además, la estación de control formada por el microcontrolador y el mando de PS2 ha permitido a los autores de este proyecto aprender a trabajar con microprocesadores y periféricos. También es una parte del proyecto que deja satisfechos a los autores del mismo. Como posibles mejoras futuras, se podría montar de manera inalámbrica la estación y el mando y conseguir un amplio rango de señal para comunicarse con el *dron*.

Para futuros proyectos (pensados en ser realizados en el TFM) se espera centrarse, como ya se ha mencionado, en el desarrollo del software de vuelo. Una vez con el objetivo claro y con una plataforma sobre la que trabajar, realizar pequeñas variaciones en ella no sería un gran problema. Otro tema sería empezar desde cero, tal como ha pasado con este proyecto. Además, se intentaría optimizar la

estación de control y, en caso que fuera posible, instalar un sistema de protección en el cuadricóptero para evitar accidentes.

Como comentario final, cabe decir que se trata de un proyecto de autoaprendizaje en el que, cumpliendo con el nombre que tiene, los autores de este proyecto y todos cuanto lo lean podrán aprender, y mucho, sobre los *drones* de uso civil. Además, los autores de este proyecto han podido desarrollar gran parte de éste aplicando conocimientos adquiridos durante su periodo de formación universitaria. Desde la mecánica del sólido rígido, álgebra y geometría para desarrollar el modelo mecánico hasta los conocimientos de programación y electrónica para montar el microprocesador y sus componentes externos.

11. Agradecimientos

Para acabar la memoria de este proyecto, a continuación los agradecimientos que merecen las personas que han apoyado a los autores del mismo.

No siempre ha sido fácil trabajar en este proyecto y al primero que quiero dar las gracias es a mi compañero de trabajo Eduard con el que he pasado tantas horas durante estos últimos cinco años y más estos últimos 6 meses. Sin él, sé que esto un hubiera sido posible. Gracias a su capacidad de sacrificio y trabajo, por tirar de mí cuando he querido abandonar y por estar siempre ahí.

Por supuesto, dar las gracias al profesor Manuel Moreno Eguilaz, cuya ayuda y guía en la realización del trabajo han sido imprescindibles. Por poner a nuestra disposición todas las facilidades con las que contaba ya fueran herramientas, un ordenador o sus consejos.

A mis hermanos Raquel y Jaume, que me han apoyado y han tenido que aguantar el mal humor con el que llegaba a casa cuando las cosas no salían. A mi padre por dar otra visión y enfoque de los problemas con los que nos hemos encontrado. A mi madre por intentar comprenderme e interesarse en lo que hacía. A los dos, en especial, por invertir en mí y en lo que hago. Gracias a Anna también por apoyarme, aconsejarme, escucharme y por sufrir conmigo.

Gracias a Albert Cerdán y Guillem Àvila del equipo del FormulaStudent ETSEIB, por echarnos una mano cuando lo hemos necesitado, por dejarnos una herramienta, por aconsejarnos y por dedicar parte de su tiempo en nosotros.

Gracias a los profesores que nos han aconsejado en algún momento cuando lo hemos necesitado.

Gracias a todos.

Por Jorge Cantero Guerrero

Llegar al final de este proyecto no ha sido un camino fácil. Durante el camino se han encontrado muchos problemas que parecían sin solución y que han llevado a muchos dolores de cabeza y frustraciones. Aguantar las horas y el trabajo solo ha sido posible por el apoyo de mi compañero y amigo Jorge, al cual doy las gracias por aguantar mis incesantes preguntas y obstinación, por frenar mis ganas de hacer sin pensar y por ser la persona con la que puedo hablar libremente, todo esto no solo durante el proyecto, sino que desde las primeras clases de Química I de hace 5 años. Se merece muchas gracias también Kumiko, que cree en mí y que me apoya a su manera para realizar mis sueños.

Gracias al profesor Manuel Moreno Eguilaz, que nos ha apoyado en la realización del proyecto, que sin su experiencia y consejo no hubiera sido posible. También doy las gracias a los padres de Jorge que nos han apoyado económicamente y así poder poner en práctica nuestras ideas.

Gracias a los chicos del FormulaStudent ETSEIB, por ayudarnos y por darnos ánimos para obtener los mejores resultados.

Por Eduard Valentino Birau.

Bibliografía

Referencias bibliográficas

[1] P.CASTILLO, P.GARCIA, R.LOZANO, P.ALBERTO; *Modelado y estabilización de un helicóptero con cuatro rotores*. Revista Iberoamericana de Automática e Informática industrial, vol. 4, núm.1, 2007.

[2] ANDREU BERNA FERRI, *desarrollo de una plataforma de tiempo real para la implementación de algoritmos de control multivariables: Ampliación al control de orientación de vehículos aéreos.* Tesina de máster, UPV, 2010.

[3] *History of quadcopters and other multirotors* - KROSSBLADE AEROSPACE. http://www.krossblade.com/history-of-quadcopters-and-multirotors/. Consultado en junio de 2015.

[4] *Oemichen 1922 - Helicopters built between 1907 and 1935.* http://www.aviastar.org/helicopters_eng/oemichen.php. Consultado en junio de 2015.

[5] *Un 'dron' español anti-incendios* - El Mundo, publicación del día 23 de noviembre de 2013. http://www.elmundo.es/ciencia/2013/11/23/528fbe7063fd3dad5a8b456f.html. Consultado en junio de 2015.

[6] *Dron de la marca HUBSAN, modelo H107*. http://www.hubsan.com/. Precio consultado en Amazon. Consultado en junio de 2015.

[7] *Dron de la marca Draganfly, modelo Guardian*. http://www.draganfly.com/uav-helicopter/draganflyer-x4p/index.php. Precio consultado en Ebay. Consultado en junio de 2015.

[8] *Arduino – Wikipedia, the free encyclopedia*. https://es.wikipedia.org/wiki/Arduino. Consultado en mayo de 2015.

[9] *Hall Effect – Wikipedia, the free encyclopedia*. https://en.wikipedia.org/wiki/Hall_effect. Consultado en junio de 2015.

[10] *Quaternion Slerp – how RTQF performs sensor fusion in the latest RTIMULib versions.* Publicación del día 29 de marzo de 2015. https://richardstechnotes.wordpress.com/2015/03/29/quaternion-slerp-how-rtqf-performs-sensor-fusion-in-the-latest-rtimulib-versions/. Consultado en junio de 2015.

[11] *Playstation 2 Controller Arduino Library V1.0 – The Mind of Bill Porter.* Publicación del día 5 de junio de 2010. http://www.billporter.info/2010/06/05/playstation-2-controller-arduino-library-v1-0/. Consultado en febrero de 2015.

[12] *PID Tuning (theory) – Autoquad documentation.* http://autoquad.org/wiki/wiki/configuring-autoquad-flightcontroller/tuning-and-troubleshooting/pid-tuning-theory/. Consultado en junio de 2015.

[13] *nRF24L01 2.4GHz Radio/Wireless Transceivers How-To – LEARN! DO! ARDUINOINFO.INFO.* http://arduino-info.wikispaces.com/Nrf24L01-2.4GHz-HowTo. Consultado en febrero de 2015.

[14] *¿Está prohibido volar drones en España? Esta es la normativa.* http://es.gizmodo.com/esta-prohibido-volar-drones-en-espana-esta-es-la-norm-1561374223. Consultado en junio de 2015.

[15] *RoHS Directive 2011/65/EU (RoHS 2) Directiva 2002/95/CE (RoHS) - Restricción de Sustancias Peligrosas. INTERNATIONAL CENTER FOR QUALITY CERTIFICATION. http://www.icqc.co.uk/es/rohs.php. Consultado en junio de 2015.*

Bibliografía complementaria

[16] MINISTERIO DE FOMENTO, GOBIERNO DE ESPAÑA; El uso de drones en España. AESA, 2014.

[17] BOE núm. 176, 23 de julio de 1960, pags. 10291 a 10299; *Ley 48/1960, 21 de julio, sobre Navegación Aérea.*

[18] *Mechatronics Project Site* – Universidad de Victoria, Canada. Desarrollado en el año 2012. http://robots.dacloughb.com/. Consultado en febrero de 2015.

[19] *The quadcopter: how to compute the pitch, roll and yaw – The Bored Engineers Blog.* http://theboredengineers.com/2012/09/the-quadcopter-get-its-orientation-from-sensors. Consultado en marzo de 2015.

[20] *How to build your own Quadcopter Autopilot / Flight Controller,* Escrito por Dr Gareth Owen. https://ghowen.me/build-your-own-quadcopter-autopilot/. Consultado en abril de 2015.

[21] *Aplicaciones y usos. – Intelligenia DYNAMICS. http://www.iuavs.com/pages/aplicaciones_y_usos*. Consultado en Junio de 2015.

[22] *El Salud factura a las aseguradoras el coste médico de los rescates de montaña.* Autor: Mª José Villanueva. Artículo publicado el día 24 de junio de 2014. http://www.heraldo.es/noticias/aragon/2014/06/24/el_salud_factura_las_aseguradoras_coste_medico_los_rescates_montana_295539_300.html. Consultado en junio de 2015.

[23] *MEMS Accelerometer – Instrumentation Today.* Publicado el día 17 de agosto de 2015. http://www.instrumentationtoday.com/mems-accelerometer/2011/08/. Consultado en mayo de 2015.

[24] *Serial Peripheral Interface – Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Serial_Peripheral_Interface. Consultada en junio de 2015.*

[25] *SPI library. https://www.arduino.cc/en/Reference/SPI. Consultada en abril de 2015.*

[26] *Network Layer for nRF24L01(+) radios.* Repositorio de la librería RF24Network publicada el 29 de setiembre de 2013. https://github.com/maniacbug/RF24Network. Consultada en el año 2015.

[27] *Network Layer for RF24 Radios, Full Documentation.* Documentación de la librería RF24Network. http://maniacbug.github.io/RF24Network/index.html. Consultada en el año 2015.

[28] *Librería PID de Arduino.* Publicada el 15 de abril de 2013. *https://github.com/strangedev/Arduino-Quadcopter/tree/master/libraries/PID_v1*. Consultada en el año 2015.

[29] *Improving the Beginner's PID – Introduction.* Publicado el 15 de abril de 2011. http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/. Consultado en el año 2015.

[30] *Playstation 2 Controller Arduino Library v1.0.* Publicada el 5 de junio de 2010. http://www.billporter.info/2010/06/05/playstation-2-controller-arduino-library-v1-0/. Consultada en el año 2015.

[31] *Arduino Playstation 2 Controller Library Troubleshooting Guide.* Publicada el 27 de marzo de 2011. http://www.billporter.info/2011/03/27/arduino-playstation-2-controller-library-troubleshooting-guide/. Consultada en el año 2015.

[32] *RTIMULib-Arduino - a versatile 9-dof and 10-dof IMU library for the Arduino. https://github.com/richards-tech/RTIMULib-Arduino*. Consultada en el año 2015.

[33] *Reciclaje de madera – Recytrans.* Publicación del 13 de setiembre de 2013. http://www.recytrans.com/blog/reciclaje-de-madera/. Consultada en junio de 2015.

[34] *El tratamiento de los RAEE´S.* http://www.limasa3.es/buenas-practicas/en-la-gestion-de-raee/el-tratamiento-de-los-raee%C2%B4s. Consultada en junio de 2015.

ANEXO

1. Desarrollo del modelo dinámico

En este primer capítulo del anexo se explica de manera más clara y concisa el desarrollo a partir de la ecuación 12 del capítulo 3.2.

$$\mathbb{J} \cdot \ddot{q} = \tau \quad \text{(Ec. 25)}$$

$$\ddot{q} = \mathbb{J}^{-1} \cdot \tau \quad \text{(Ec. 26)}$$

En la que cada término corresponde a las coordenadas generalizadas, la inversa de la matriz de inercia en función de una orientación general y los momentos de cada eje.

$$\ddot{q} = \begin{pmatrix} \ddot{\theta} \\ \ddot{\varphi} \\ \ddot{\psi} \end{pmatrix} \quad \text{(Ec. 27)}$$

$$\tau = \begin{pmatrix} M_\theta \\ M_\varphi \\ M_\psi \end{pmatrix} \quad \text{(Ec. 28)}$$

El desarrollo de la matriz $\mathbb{J}^{-1}$se explica a continuación:

Tras la simplificación por serie de Taylor de primer grado de las matrices de rotación para ángulos pequeños, dónde:

$$\cos\alpha \approx 1 \quad \text{(Ec. 29)}$$
$$\sin\alpha \approx \alpha \quad \text{(Ec. 30)}$$

La matriz de rotación del *eje x*, correspondiente al ángulo de pitch:

$$R_x^\theta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -\theta \\ 0 & \theta & 1 \end{bmatrix} \quad \text{(Ec. 31)}$$

La matriz de rotación del *eje y*, correspondiente al ángulo de roll:

$$R_y^\varphi = \begin{bmatrix} 1 & 0 & \varphi \\ 0 & 1 & 0 \\ -\varphi & 0 & 1 \end{bmatrix} \quad \text{(Ec. 32)}$$

La matriz de rotación del *eje z*, correspondiente al ángulo de yaw:

$$R_z^\psi = \begin{bmatrix} 1 & -\psi & 0 \\ \psi & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{(Ec. 33)}$$

Considerando al cuadricóptero simétrico, el tensor de inercia del dron en la referencia descrita en la Fig. 11:

$$\mathbb{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \quad \text{(Ec. 34)}$$

El giro general, resultado del producto de las rotaciones respecto cada eje:

$$R_z^{\psi} \cdot R_y^{\varphi} \cdot R_x^{\theta} = \begin{bmatrix} 1 & \theta\varphi - \psi & \varphi + \theta\psi \\ \psi & \psi\varphi\theta + 1 & \varphi\psi - \theta \\ -\varphi & \theta & 1 \end{bmatrix} \quad \text{(Ec. 35)}$$

Tras menospreciar el producto de ángulos pequeños queda:

$$R = \begin{bmatrix} 1 & -\psi & \varphi \\ \psi & 1 & -\theta \\ -\varphi & \theta & 1 \end{bmatrix} \quad \text{(Ec. 36)}$$

Se ha definido la matriz $\mathbb{J}$ como:

$$\mathbb{J} = R^t \mathbb{I}\, R = \quad \text{(Ec. 37)}$$
$$= \begin{bmatrix} I_{zz}\varphi^2 + I_{yy}\psi^2 + I_{xx} & I_{yy}\psi - I_{xx}\psi - I_{zz}\varphi\theta & I_{xx}\varphi - I_{zz}\varphi - I_{yy}\psi\theta \\ I_{yy}\psi - I_{xx}\psi - I_{zz}\varphi\theta & I_{zz}\theta^2 + I_{xx}\psi^2 + I_{yy} & I_{zz}\theta - I_{yy}\theta - I_{xx}\psi\varphi \\ I_{xx}\varphi - I_{zz}\varphi - I_{yy}\psi\theta & I_{zz}\theta - I_{yy}\theta - I_{xx}\psi\varphi & I_{yy}\theta^2 + I_{xx}\varphi^2 + I_{zz} \end{bmatrix}$$

Que, una vez simplificada (se vuelve a considerar el producto de variables de pequeña señal como nulos):

$$\mathbb{J} = \begin{bmatrix} I_{xx} & (I_{yy} - I_{xx})\psi & (I_{xx} - I_{zz})\varphi \\ (I_{yy} - I_{xx})\psi & I_{yy} & (I_{zz} - I_{yy})\theta \\ (I_{xx} - I_{zz})\varphi & (I_{zz} - I_{yy})\theta & I_{zz} \end{bmatrix} \quad \text{(Ec. 38)}$$

Cuya inversa es:

$$\mathbb{J}^{-1} = \begin{bmatrix} \frac{1}{I_{xx}} & \frac{(I_{zz}^2 - I_{yy}I_{zz})}{I_{xx}I_{yy}I_{zz}}\psi & \frac{(I_{yy}I_{zz} - I_{xx}I_{yy})}{I_{xx}I_{yy}I_{zz}}\varphi \\ \frac{(I_{zz}^2 - I_{yy}I_{zz})}{I_{xx}I_{yy}I_{zz}}\psi & \frac{1}{I_{yy}} & \frac{(I_{xx}I_{yy} - I_{xx}I_{zz})}{I_{xx}I_{yy}I_{zz}}\theta \\ \frac{(I_{yy}I_{zz} - I_{xx}I_{yy})}{I_{xx}I_{yy}I_{zz}}\varphi & \frac{(I_{xx}I_{yy} - I_{xx}I_{zz})}{I_{xx}I_{yy}I_{zz}}\theta & \frac{1}{I_{zz}} \end{bmatrix} \quad \text{(Ec. 39)}$$

En cuanto al vector de momentos de la ecuación26, queda definido como:

$$\tau = \begin{bmatrix} M_\theta \\ M_\varphi \\ M_\psi \end{bmatrix} = \begin{bmatrix} (T_3 - T_1) \cdot d \\ (T_4 - T_2) \cdot d \\ -\frac{K_{drag}}{K_{empuje}} \cdot \sum_{i=1}^{4} T_i \end{bmatrix} = \quad \text{(Ec. 40)}$$
$$= \begin{bmatrix} -d & 0 & d & 0 \\ 0 & -d & 0 & d \\ -\frac{K_{drag}}{K_{empuje}} & -\frac{K_{drag}}{K_{empuje}} & -\frac{K_{drag}}{K_{empuje}} & -\frac{K_{drag}}{K_{empuje}} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix}$$

Finalmente, al multiplicar $\mathbb{J}^{-1}$por τ queda:

$$\ddot{\eta} = \mathbb{J}^{-1} \cdot \tau = \begin{bmatrix} \frac{M_\theta}{I_{xx}} + \frac{(I_{zz}^2 - I_{yy}I_{zz})}{I_{xx}I_{yy}I_{zz}}\psi M_\varphi + \frac{(I_{yy}I_{zz} - I_{xx}I_{yy})}{I_{xx}I_{yy}I_{zz}}\varphi M_\psi \\ \frac{(I_{zz}^2 - I_{yy}I_{zz})}{I_{xx}I_{yy}I_{zz}}\psi M_\theta + \frac{M_\varphi}{I_{yy}} + \frac{(I_{xx}I_{yy} - I_{xx}I_{zz})}{I_{xx}I_{yy}I_{zz}}\theta M_\psi \\ \frac{(I_{yy}I_{zz} - I_{xx}I_{yy})}{I_{xx}I_{yy}I_{zz}}\varphi M_\theta + \frac{(I_{xx}I_{yy} - I_{xx}I_{zz})}{I_{xx}I_{yy}I_{zz}}\theta M_\varphi + \frac{M_\psi}{I_{zz}} \end{bmatrix} \quad \text{(Ec. 41)}$$

Esta última expresión es la que finalmente se ha usado para implementar el modelo en Matlab.

2. Planos de las piezas del cuadricóptero

En este capítulo del anexo se encuentran los planos de croquis de las piezas que los autores de este proyecto han tenido que conformar para poder construir el *dron*. En el capítulo 4 se explica el porqué de su diseño y la elección del material. A continuación se aprecian los croquis con las cotas básicas que pueden servir de guía para emprendedores que quieran fabricar su propio *dron*. Se deja libertad de pequeñas variaciones en las cotas para conseguir el ensamblaje de todas las partes ya que, en función de los medios que se disponga, no resulta sencillo conseguirlo.

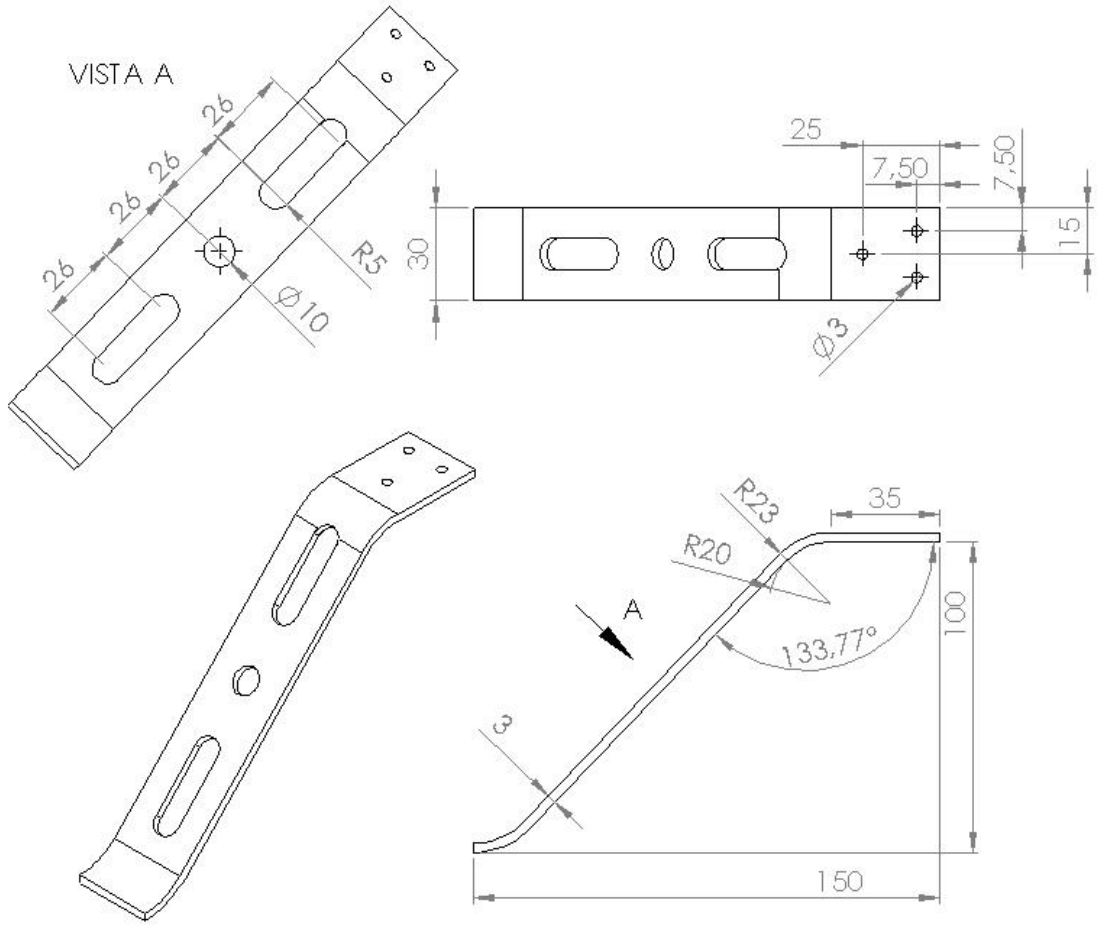

Fig. 101. Croquis de una pata de aluminio. (Fuente: propia)

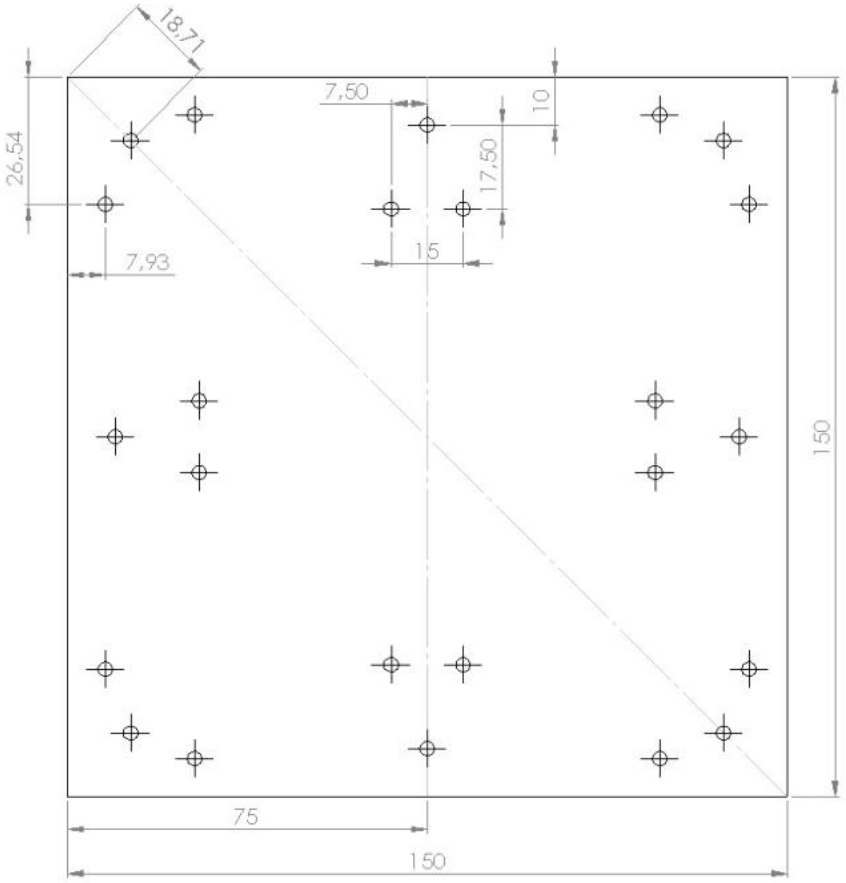

Fig. 102. Croquis de la plataforma central inferior de madera. (Fuente: propia)

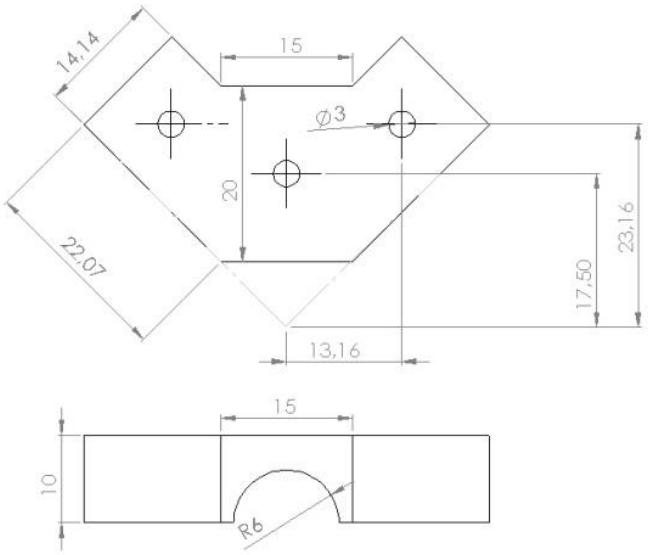

Fig. 103. Croquis de la sujeción impresa 3D ABS. (Fuente: propia)

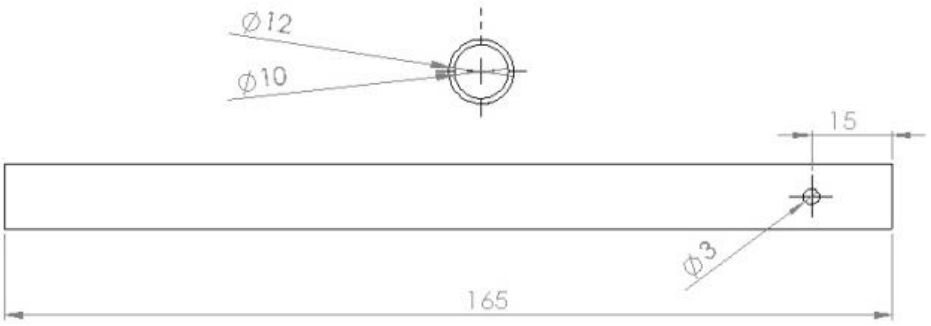

Fig. 104. Croquis brazo de metacrilato. (Fuente: propia)

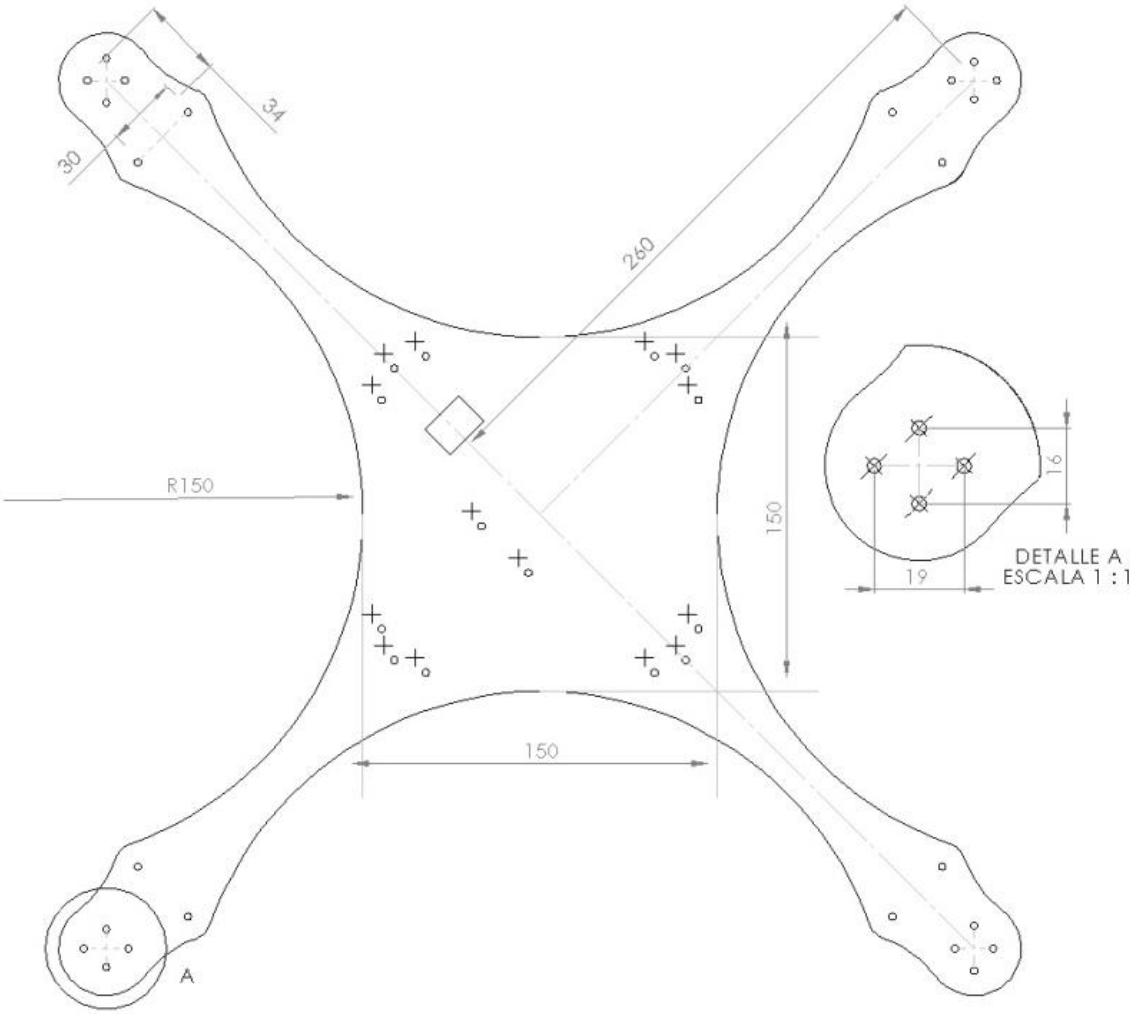

Fig. 105. Croquis plataforma central superior de madera. (Fuente: propia)

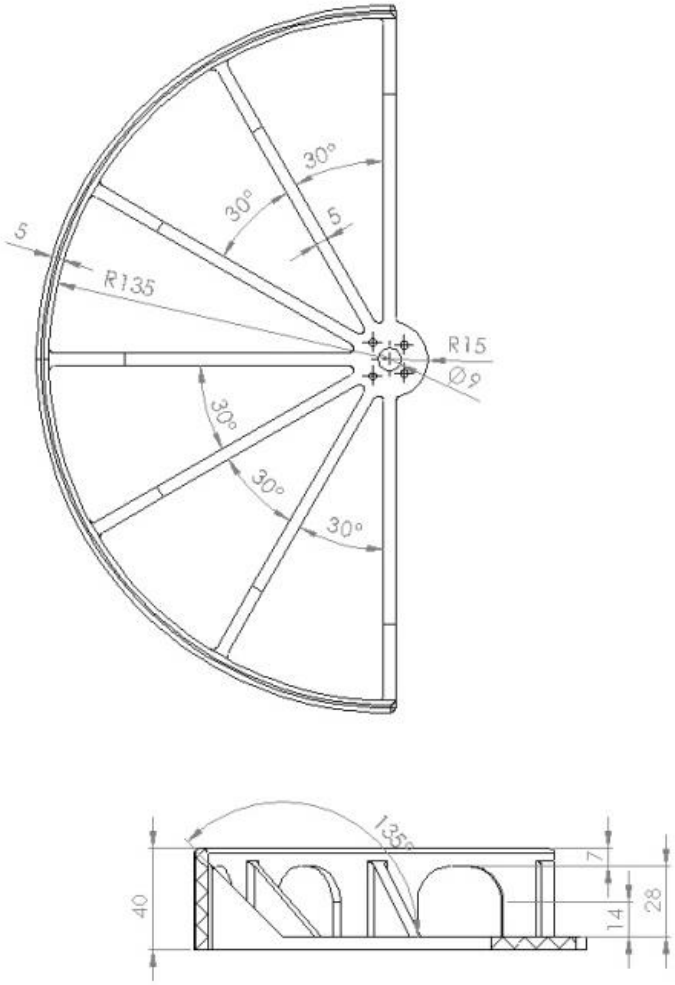

Fig. 106. Croquis de las protecciones de hélices impresas 3D de ABS. (Fuente: propia)

3. Código fuente Arduino

3.1. Código fuente del mando de radiocontrol

3.1.1. Archivo RF_TX.ino

```
//Importa librerias para el mando de la Play2, para el bus SPI, y las libre
rias para utilizar el modulo NRF24L01//

#include <RF24Network.h>
#include <RF24Network_config.h>
#include <Sync.h>

#include <PS2X_lib.h>
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"

#include "Config.h"

//Estructura del menasje
struct MyData {
  byte ide;
  byte throttle = 0;
  byte yaw = 127;
  byte pitch = 127;
  byte roll = 127;
  byte armado = 0;
  byte CRC;
};

MyData data;
//Crea  el objeto radio con la señal CE y CSN en los pins 9 y 53 respectiva
mente
RF24 radio(9, 53);
RF24Network network(radio);
// Direccion de este nodo
const uint16_t this_node = 1;

// Dirección del otro nodo
const uint16_t other_node = 0;
//Crea el objeto mando PS2
PS2X ps2x;
int time;
void setup()
{
  SPI.begin(); //inicializa el objeto radio y abre el canal para envio de
datos
  radio.begin();
  network.begin(/*channel*/ 90, /*node address*/ this_node);
  delay(1000);

  bool error = ps2x.config_gamepad(PS2_CLK, PS2_CMD, PS2_SEL, PS2_DAT,
true, true); //configura el objeto mando PS2 en los pines correspondientes
  senyal_Armado_Mot(); //Arma los motores
  delay(2000);
}
void loop()
{
  ps2x.read_gamepad(false, 0); //Lee el mando
```

```
  int i = 0;
  while ( i <= 2) {
    if (ps2x.NewButtonState(PSB_L3)) {
      data.armado = 127;
    }
    else {
      data.armado = 0;
    }
    i += 1;
  }
  data.ide = 0x55;
  data.throttle = ps2x.Analog(PSS_LY);
  data.yaw = ps2x.Analog(PSS_LX);
  data.pitch = ps2x.Analog(PSS_RY);
  data.roll = ps2x.Analog(PSS_RX);
  const byte joystick[5] = {data.throttle, data.yaw, data.pitch, data.roll,
data.armado};
  byte CRC1 = CRC8(joystick, sizeof(joystick));
  data.CRC = CRC1;
  network.update();
  RF24NetworkHeader header(/*to node*/ other_node);
  network.write(header, &data, sizeof(MyData)); //Envia el array joystick
al otro modulo
}
```

3.1.2. Archivo Armado_motores.ino

```
void senyal_Armado_Mot(){
  bool inizializado=false;
  while (!inizializado) {//hasta que no se aprete el boton L3 no saldra del
bucle y no configurara los motores
    if (ps2x.NewButtonState(PSB_L3)){
      data.armado=255;
      network.update();
      RF24NetworkHeader header(/*to node*/ other_node);
      network.write(header,&data,sizeof(MyData));
      inizializado=true;
      }
    ps2x.read_gamepad(false,0);
  }
}
```

3.1.3 Archivo Config.h

```
//Define los pines a los que irá conectado el mando PS2
#define PS2_DAT  13
#define PS2_CMD  11
#define PS2_SEL  10
#define PS2_CLK  12
```

3.1.4 Archivo CRC-8.ino

```
//CRC-8 - based on the CRC8 formulas by Dallas/Maxim
//code released under the therms of the GNU GPL 3.0 license
byte CRC8(const byte *data, byte len) {
  byte crc = 0x00;
  while (len--) {
    byte extract = *data++;
    for (byte tempI = 8; tempI; tempI--) {
      byte sum = (crc ^ extract) & 0x01;
      crc >>= 1;
```

```
      if (sum) {
        crc ^= 0x8C;
      }
      extract >>= 1;
    }
  }
  return crc;
}
```

3.2. Código fuente del controlador de vuelo

3.2.1 Archivo RX_NETWORK_IMU_PID.ino

```
#include "Config.h"
//Libreria PID
#include <PID_v1.h>
//Librerias sensor
#include <Wire.h>
#include "I2Cdev.h"
#include "RTIMUSettings.h"
#include "RTIMU.h"
#include "RTFusionRTQF.h"
#include "CalLib.h"
#include <EEPROM.h>
//Librerias radio
#include <RF24Network.h>
#include <RF24Network_config.h>
#include <Sync.h>
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"

//Objeto sensor
RTIMU *imu;                                           // the IMU object
RTFusionRTQF fusion;                                  // the fusion object
RTIMUSettings settings;                               // the settings
object

//Objeto radio. Crea  el objeto radio con la señal CE y CSN en los pins 9 y
 53 respectivamente
RF24 radio(9, 53);
// Crea el objeto network
RF24Network network(radio);
// Dirección logica de este nodo
const uint16_t this_node = 0;
// Dirección logica del otro nodo
const uint16_t other_node = 1;

//Parametros de adecuación de señales
float integral = 0;
float integral_yaw = 0;

//Parametros del vuelo
float yaw = 0;
float throttle = 700;
float pitch = 0;
float roll = 0;

//Estructura del mensaje
struct MyData {
  byte ide;
  byte throttle = 0;
  byte yaw = 127;
  byte pitch = 127;
  byte roll = 127;
  byte armado = 0;
  byte CRC;
};
MyData data;
byte CRC1;
```

```
//Define el array donde se guardaran las señales a enviar a los motores
float senyals[] = {0, 0, 0, 0};
//Define un array que contiene los pines a los que se conectaran los motore
s del 1 al 4 respectivamente
const int MOTOR_PIN[] = {3, 5, 6, 7};

//Parametros del sensor y del PID
float set_roll, set_pitch;
float pitch_gyro, roll_gyro;
float pitch_imu, roll_imu;
float yaw_imu;
float PIDroll_val, PIDpitch_val;
bool flag = false;

//Configuración PID
#ifdef ANGULO
PID PIDangleX(&roll_imu, &set_roll, &roll, ANGLEX_KP, ANGLEX_KI, ANGLEX_KD,
DIRECT);
PID PIDangleY(&pitch_imu, &set_pitch, &pitch, ANGLEY_KP, ANGLEY_KI,
ANGLEY_KD, DIRECT);
#endif
PID PIDroll(&roll_gyro, &PIDroll_val, &set_roll, ROLL_PID_KP, ROLL_PID_KI,
ROLL_PID_KD, REVERSE);
PID PIDpitch(&pitch_gyro, &PIDpitch_val, &set_pitch, PITCH_PID_KP,
PITCH_PID_KI, PITCH_PID_KD, REVERSE);
int kp = 0;
int ki = 0;
int kd = 0;

void setup()
{
#ifdef DEBUG
  //Inicialización Serial
  Serial.begin(SERIAL_PORT_SPEED);
#endif
  //Inizializacion radio
  SPI.begin();
  radio.begin();
  network.begin(90, this_node);
  //Armado motores
  armado_Mot();
  //Inizialiacion sensor
  set_IMU();
  //Inizialización PID
  set_PID();
}

void loop() {
  int loopCount = 1;
  while (imu->IMURead()) { //Lectura de los ultimos datos del sensor
    if (++loopCount >= 10) {
      continue;
    }

    //Lectura datos recibidos por RF
    network.update();
    if ( network.available() ) {
      bool done = false;
      while (!done) {
        RF24NetworkHeader header;
```

```
        done = network.read(header, &data, sizeof(MyData));
      }
    }

    //Calculo de la nueva orientación
    fusion.newIMUData(imu->getGyro(), imu->getAccel(), imu-
>getCompass(), imu->getTimestamp());
  }

  //Variación de los angulos pitch y roll [deg/s]
  pitch_gyro = imu->getGyro().y() * RTMATH_RAD_TO_DEGREE;
  roll_gyro = imu->getGyro().x() * RTMATH_RAD_TO_DEGREE;
  // Angulos pitch y roll [deg]
  pitch_imu = fusion.getFusionPose().y() * RTMATH_RAD_TO_DEGREE;
  roll_imu = fusion.getFusionPose().x() * RTMATH_RAD_TO_DEGREE;
  //yaw_imu = fusion.getFusionPose().z() * RTMATH_RAD_TO_DEGREE;

  //Control del mensaje recibido
  if (data.ide == 0x55) {
    const byte joystick[5] = {data.throttle, data.yaw, data.pitch,
data.roll, data.armado};
    CRC1 = CRC8(joystick, sizeof(joystick));
    if (CRC1 == data.CRC) {
      if (data.armado == 127) { //Desarmado de motores
        throttle = 700;
        for (int i = 0; i <= 3; ++i) {
          write_Mot(MOTOR_PIN[i] ,  throttle);
        }
        while (1) {}
      }
      else {
        throttle = lectura_Throttle(data.throttle);
        //yaw = lectura_Yaw(data.yaw);
        pitch = -lectura_Pitch(data.pitch);
        roll = -lectura_Roll(data.roll);
      }
    }
  }

  //Limitación de la variable throttle
  if (throttle > 1600) {
    throttle = 1600;
  }
  if (throttle > 1200) {
#ifdef ANGULO
    PIDangleX.Compute();
    PIDangleY.Compute();
#endif
#ifdef VELOCIDAD
    roll = set_roll;
    pitch = set_pitch;
#endif
    PIDroll.Compute();
    PIDpitch.Compute();
#ifdef DEBUG
    Serial.print(throttle); Serial.print(" ");
    Serial.print(roll); Serial.print(" ");
    Serial.print(pitch); Serial.print(" ");
    //Serial.print(yaw);Serial.println(" ");
    //Serial.print(yaw_imu);Serial.println(" ");
    Serial.print(roll_imu); Serial.print(" ");
```

```
    Serial.print(pitch_imu); Serial.print(" ");
    Serial.print(roll_gyro); Serial.print(" ");
    Serial.print(pitch_gyro); Serial.print(" ");
    Serial.print(set_roll); Serial.print(" ");
    Serial.print(set_pitch); Serial.print(" ");
    Serial.print(PIDroll_val); Serial.print(" ");
    Serial.print(PIDpitch_val); Serial.print(" ");
    Serial.print(PIDroll.GetKp()); Serial.print(" ");
    Serial.print(PIDroll.GetKi()); Serial.print(" ");
    Serial.print(PIDroll.GetKd()); Serial.println(" ");
    Serial.print(PIDpitch.GetKp()); Serial.print(" ");
    Serial.print(PIDpitch.GetKi()); Serial.print(" ");
    Serial.print(PIDpitch.GetKd()); Serial.println(" ");
#endif
    //A partir de los parametros de vuelo calcula la senyal a enviar a los
motores y los envia
    senyals[0] = throttle - PIDpitch_val /*+ yaw*/;
    senyals[1] = throttle + PIDroll_val /*- yaw*/;
    senyals[2] = throttle + PIDpitch_val /*+ yaw*/;
    senyals[3] = throttle - PIDroll_val /*- yaw */;
#ifdef DEBUG
    Serial.print(senyals[0]); Serial.print(" ");
    Serial.print(senyals[1]); Serial.print(" ");
    Serial.print(senyals[2]); Serial.print(" ");
    Serial.print(senyals[3]); Serial.println(" ");
#endif
    for (int i = 0; i <= 3; ++i) {
      write_Mot(MOTOR_PIN[i] , senyals[i]);
    }
  }
  else {
    for (int i = 0; i <= 3; ++i) {
      write_Mot(MOTOR_PIN[i] , throttle);
    }
  }

  //readAndChange();
}
```

3.2.2. Archivo PID.ino

```
//Void para inizializacion
void set_PID() {
#ifdef ANGULO
  PIDangleX.SetMode(AUTOMATIC);
  PIDangleX.SetSampleTime(10);
  PIDangleX.SetOutputLimits(ANGLEX_MIN, ANGLEX_MAX);
  PIDangleY.SetMode(AUTOMATIC);
  PIDangleY.SetSampleTime(10);
  PIDangleY.SetOutputLimits(ANGLEX_MIN, ANGLEX_MAX);
#endif
  PIDroll.SetMode(AUTOMATIC);
  PIDroll.SetSampleTime(10);
  PIDroll.SetOutputLimits(ROLL_PID_MIN, ROLL_PID_MAX);
  PIDpitch.SetMode(AUTOMATIC);
  PIDpitch.SetSampleTime(10);
  PIDpitch.SetOutputLimits(PITCH_PID_MIN, PITCH_PID_MAX);
}

void readAndChange() {
  if (Serial.available() > 0) {
```

```
    char a = Serial.read();
    a = a - '0';
    switch (a) {
      case 0:
        kp = kp + 50;
        break;
      case 1:
        kp = kp - 50;
        break;
      case 2:
        ki = ki + 5;
        break;
      case 3:
        ki = ki - 5;
        break;
      case 4:
        kd = kd + 1;
        break;
      case 5:
        kd = kd - 1;
        break;
    }
    //PIDroll.SetTunings((float)kp/1000.0,(float)ki/1000.0,(float)kd/1000.0
);
    PIDpitch.SetTunings((float)kp / 1000.0, (float)ki / 1000.0, (float)kd /
1000.0);
  }
}
```

3.2.3. Archivo MOT.ino

```
//Void para enviar señal a los motores
void write_Mot(int mot, float microSeconds) {
  microSeconds = map(microSeconds, 0, 2000, 0, 255);
  analogWrite(mot, microSeconds);
}

//Void para armar los motores
void armado_Mot() {
  //Inicializacion pines motores
  for (int i = 0; i <= 3; ++i) {
    pinMode(MOTOR_PIN[i], OUTPUT);
  }
  bool inicializado = false;
  while (!inicializado) {
    bool done = false;
    network.update();
    if ( network.available() ) {
      while (!done) {
        RF24NetworkHeader header;
        network.read(header, &data, sizeof(MyData));
        if (data.armado == 255) {
          done = true;
        }
      }
    }
    if (done) {
      for ( int i = 0; i <= 3; ++i) {
        write_Mot(MOTOR_PIN[i] , MIN_SIGNAL); // envia la señal minima a
los motores para que se armen
      }
```

```
      inicializado = true;
    }
  }
#ifdef DEBUG
  Serial.println("motores armados");
#endif
}
```

3.2.4. Archivo IMU.ino

```
void set_IMU() {
  int errcode;
  Wire.begin();
  imu = RTIMU::createIMU(&settings);                        // create the
imu object
#ifdef DEBUG
  Serial.print("ArduinoIMU starting using device "); Serial.println(imu-
>IMUName());
#endif
  if ((errcode = imu->IMUInit()) < 0) {
#ifdef DEBUG
    Serial.print("Failed to init IMU: "); Serial.println(errcode);
#endif
  }

  if (imu->getCalibrationValid())
#ifdef DEBUG
    Serial.println("Using compass calibration");
#endif
  else
#ifdef DEBUG
    Serial.println("No valid compass calibration data");
#endif
  fusion.setSlerpPower(0.02);
  fusion.setGyroEnable(true);
  fusion.setAccelEnable(true);
  fusion.setCompassEnable(true);
  imu->IMURead();
}
```

3.2.5. Archivo adec_Pitch.ino

```
float lectura_Pitch(byte value_RY){
    if (value_RY < acc_min){
      pitch = int_pitch_MAX*(1 - value_RY/acc_min);
    }
    else if (value_RY > fren_min){
      pitch = (int_pitch_MAX/(255-fren_min))*(fren_min-value_RY);
    }
    else if (acc_min <= value_RY <= fren_min){
      pitch = 0;
    }
  return pitch;
}
```

3.2.6. Archivo adec_Roll.ino

```
float lectura_Roll (byte value_RX){
    if (value_RX < acc_min){
      roll = int_roll_MAX*(1 - value_RX/acc_min);
    }
```

```
    else if (value_RX > fren_min){
      roll = (int_roll_MAX/(255-fren_min))*(fren_min-value_RX);
    }
    else if (acc_min <= value_RX <= fren_min){
      roll = 0;
    }
  return roll;
}
```

3.2.7. Archivo adec_Yaw.ino

```
float lectura_Yaw(byte value_LX){
   if (value_LX < acc_min){
      integral_yaw = int_yaw_MAX*(1 - value_LX/acc_min);
  }
  else if (value_LX > fren_min){
    integral_yaw = (int_yaw_MAX/(255-fren_min))*(fren_min-value_LX);
  }
  else if (acc_min <= value_LX <= fren_min){
    integral_yaw = 0;

  }
  if (integral_yaw>0){
    yaw=yaw_imu+integral_yaw;
    flag=false;
  }
  else if(integral_yaw<0){
    yaw=yaw_imu-integral_yaw;
    flag=false;
  }
  if(flag==false&&integral_yaw==0){
    yaw=yaw_imu;
    flag=true;
    }
  return yaw;
}
```

3.2.8. Archivo adec_Throttle.ino

```
float lectura_Throttle(byte value_LY){
  if (value_LY < acc_min){
      integral = int_MAX*(1 - value_LY/acc_min);
  }
  else if (value_LY > fren_min){
    integral = (int_MAX/(255-fren_min))*(fren_min-value_LY);
  }
  else if (acc_min <= value_LY <= fren_min){
    integral = 0;
  }
  if (integral>0&&throttle+integral<MAX_SIGNAL ){
    throttle+=integral;
  }
  else if(integral<0&&throttle+integral>MIN_SIGNAL){
    throttle+=integral;
  }
  return throttle;
}
```

3.2.9. Archivo Config.h

```
#define ANGLEX_KP 0.0
```

```
#define ANGLEX_KI 0.0
#define ANGLEX_KD 0.0
#define ANGLEX_MIN -50
#define ANGLEX_MAX 50

#define ANGLEY_KP 0.0
#define ANGLEY_KI 0.0
#define ANGLEY_KD 0.0
#define ANGLEY_MIN -50
#define ANGLEY_MAX 50

#define ROLL_PID_KP 0.95
#define ROLL_PID_KI 1.14
#define ROLL_PID_KD 0.007
#define ROLL_PID_MIN -200
#define ROLL_PID_MAX 200

#define PITCH_PID_KP 0.95
#define PITCH_PID_KI 1.14
#define PITCH_PID_KD 0.007
#define PITCH_PID_MIN -200
#define PITCH_PID_MAX 200

#define MAX_SIGNAL 2000
#define MIN_SIGNAL 700

#define MAX_yaw 180.0
#define MIN_yaw -180.0

#define int_pitch_MAX 60.0
#define int_roll_MAX 60.0
#define int_yaw_MAX 2.0

#define int_MAX  10.0
#define acc_min 100.0
#define fren_min 140.0

#define  SERIAL_PORT_SPEED  115200
#define DEBUG
//#define ANGULO
#define VELOCIDAD
```

3.2.10. Archivo CRC-8.ino

```
//CRC-8 - based on the CRC8 formulas by Dallas/Maxim
//code released under the therms of the GNU GPL 3.0 license
byte CRC8(const byte *data, byte len) {
  byte crc = 0x00;
  while (len--) {
    byte extract = *data++;
    for (byte tempI = 8; tempI; tempI--) {
      byte sum = (crc ^ extract) & 0x01;
      crc >>= 1;
      if (sum) {
        crc ^= 0x8C;
      }
      extract >>= 1;
    }
  }
  return crc;
}
```

4. Resultados de las pruebas de los motores

Throttle (PWM)	Hz	RPM	rad/s	(rad/s)^2	Empuje (g)	Empuje(N)	Par1(g)	Par2(g)	Par(Nm)
700	0	0	0,0000	0,0000	0	0,0000	0	0	0,0000
750	32	960	100,5310	10106,4746	7	0,0687	0	1	0,0010
800	49	1470	153,9380	23696,9194	19	0,1864	4	6	0,0103
850	65	1950	204,2035	41699,0772	35	0,3434	8	10	0,0185
900	79	2370	248,1858	61596,1990	52	0,5101	15	19	0,0350
950	90	2700	282,7433	79943,7929	70	0,6867	17	20	0,0381
1000	101	3030	317,3009	100679,8311	90	0,8829	19	24	0,0443
1050	111	3330	348,7168	121603,3917	108	1,0595	23	26	0,0505
1100	120	3600	376,9911	142122,2985	130	1,2753	27	24	0,0525
1150	132	3960	414,6902	171967,9812	158	1,5500	31	28	0,0608
1200	143	4290	449,2477	201823,5335	185	1,8149	33	30	0,0649
1250	156	4680	490,0884	240186,6845	222	2,1778	40	37	0,0793
1300	166	4980	521,5044	271966,8096	262	2,5702	45	42	0,0896
1350	176	5280	552,9203	305720,8555	295	2,8940	45	46	0,0937
1400	188	5640	590,6194	348831,2861	330	3,2373	50	53	0,1061
1450	199	5970	625,1769	390846,1906	370	3,6297	54	59	0,1164
1500	209	6270	656,5929	431114,1751	410	4,0221	57	61	0,1215
1550	220	6600	691,1504	477688,8367	460	4,5126	57	69	0,1298
1600	230	6900	722,5663	522102,0550	490	4,8069	60	82	0,1463
1650	238	7140	747,6990	559053,8526	540	5,2974	65	84	0,1535
1700	248	7440	779,1150	607020,1284	600	5,8860	70	88	0,1627

Tabla 15. Resultados ensayos con motores. (Fuente: propia)

5. Normativa

Proyectos relacionados con el diseño y la fabricación de vehículos, ya sean terrestres, marítimos o aéreos, que incluyen novedades tecnológicas o se alejan de los estándares conyevan conflictos normativos que hay que tener en cuenta. Es el caso de este proyecto.

Al ser los cuadricópteros (o *drones*, conocidos popularmente) algo novedoso en la sociedad, todavía surgen dudas en cuanto a qué se puede y no se puede hacer con ellos. Otro tema es el sector militar, en el que no se pretende entrar en este proyecto.

A continuación se cita un extracto del blog de Carlos Zahumensky [14] en el que explica de manera llana y clara los límites legales que hay que respetar cuando se maneja un *dron*.

> "*La Agencia Estatal de Seguridad Aérea (AESA) ha publicado una nota aclaratoria que puede haber pillado por sorpresa a más de uno. El documento [16] no refleja nada nuevo, pero recuerda un detalle que quizá no era muy conocido entre el público: los drones no pueden volar a su antojo en España. Hay unas normas, y son bastante restrictivas.*
>
> ***¿Qué es un dron?***
>
> *La pregunta parece de perogrullo, pero es importante a nivel legal. La Agencia considera que* ***solo es un dron cuando tiene fines comerciales o profesionales****. En otras palabras, que si nos compramos un Parrot AR Drone o un DJI Phantom para filmar nuestro pueblo, desde el punto de vista legal no es un dron (aunque para nosotros sí lo sea).*
>
> *Sin embargo, si vendemos esa grabación a, por ejemplo, una oficina de turismo, o la colgamos en nuestro blog con publicidad sí que lo es, porque lo estamos usando con fines comerciales. En este sentido, un helicóptero de radiocontrol utilizado para hacer tomas aéreas también es un dron.*
>
> ***Una paradoja legal***
>
> *El problema actual es que los drones son tan nuevos que están en una especie de limbo legal. Actualmente, se las considera aeronaves, por lo que están sujetos a la normativa general redactada en los artículos 150 y 151 de la Ley 48/1960 [17] sobre Navegación Aérea.*
>
> *En teoría, para poder volar drones con fines comerciales, la ley obliga a pedir permiso a AESA, pero se da la circunstancia de que AESA no puede emitir este permiso porque no hay una normativa específica que ofrezca una base legal. La Agencia está redactando esta normativa junto al Ministerio de Industria. Hasta que no llegue, no pueden autorizar nada.*
>
> *Para complicar el embrollo, hay comunidades autónomas que tienen sus propios reglamentos sobre drones. AESA recomienda a las personas que vayan a volar estos dispositivos que se pongan en contacto con su ayuntamiento para consultar si hay alguna norma local al respecto.*
>
> ***Utilizar drones con fines lúdicos***

Llegamos al punto que la interesa a la mayoría. Si simplemente vamos a pilotar un dron para divertirnos, legalmente no es un dron, sino un vehículo de radiocontrol, y está bajo la jurisdicción de la Real Federación Aeronáutica de España y cada comunidad autónoma.

En este caso también hay restricciones. El aparato no puede volar a altitudes superiores a los 100 metros de altura, ni sobre núcleos urbanos o lugares habitados. De hacerlo, podría enfrentarse a sanciones si alguien lo denuncia. […] El vuelo en interiores, al no formar parte del espacio aéreo, no está regulado.

El uso de drones con fines lúdicos y deportivos también será regulado en la nueva legislación específica.[…]"